CA ArcotID® PKI

Client Reference Guide

r6.2.0.2

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Introduction

This guide is designed to be a reference manual for you as you create custom applications that use CA ArcotID PKI Client for authentication and digital signatures.

This chapter discusses the following topics:

Note: CA ArcotID PKI Client still contains the terms Arcot, WebFort, and ArcotID in some of its code objects and other artifacts. Therefore, you will find occurrences of Arcot, WebFort, and ArcotID in all CA ArcotID PKI Client documentation. In addition, some of the topics in this guide do not follow the standard formatting guidelines. These inconsistencies will be fixed in a future release.

ArcotID PKI and ArcotID PKI Client

ArcotID PKI is a *secure software credential* that provides two-factor authentication and protects digital IDs using *cryptographic camouflage* technology.

An ArcotID PKI is a small data file that by itself can be used for strong authentication to a variety of applications such as Web or Virtual Private Networks (VPNs). ArcotID PKI is not vulnerable to "brute force" password attacks or "man-in-the-middle" attacks.

The size of the ArcotID PKI depends on the presence of the logo image. If the logo is present, then the ArcotID PKI size can range between 3KB-100KB, otherwise it is typically between 2KB-3KB.

When an ArcotID PKI is used to protect one or more Digital IDs, it provides the same features as a physical smartcard or USB security token such as, client-server strong authentication, digital signing, encryption, and decryption. The ArcotID PKI Client software provides standard CSP and PKCS#11 protocol support for PKI-enabled applications.

Note: To get a better understanding of how to integrate the SDK with your client application, see the sample application shipped with the client package described in this guide.

Features and Usage Scenarios

This section describes the important features of ArcotID PKI Client and the usage scenarios.

- **Web and VPN authentication**

  Web applications and Web-based VPN requiring authentication can use the ArcotID PKI Client to provide two-factor strong authentication. The ArcotID PKI Client is available as a browser plugin, java applet, and flash client. This wide variety of deployment options enables the ArcotID PKI Client to be used in most Web browsers and operating systems.

- **SSL client authentication using a digital ID in the ArcotID PKI**

  The optional ArcotID PKI *key vault* feature enables the ArcotID PKI to protect additional *private key* portion of the digital ID. When the key vault feature is used, the ArcotID PKI Client can be used for SSL client authentication with the help of CA's Cryptographic Service Provider (CSP) or Arcot PKCS#11 module.

- **Digital signing by using a digital ID in the ArcotID PKI**

  The ArcotID PKI key vault feature also enables the digital IDs stored in an ArcotID PKI to be used for digital signing in PKI-enabled applications such as email, PDF files, and other document types. The ArcotID PKI Client signing functionality is supported for most applications that support PKCS#11 and Microsoft cryptographic API standards.

- **Decrypting PKI encrypted files or emails by using a digital ID that is stored in the ArcotID PKI**

  The ArcotID PKI key vault feature enables the ArcotID PKI Client to protect the decryption keys, which are used to decrypt the encrypted emails, PDF files, and other documents. The ArcotID PKI Client decryption functionality is supported for most applications that support the PKCS#11 and Microsoft cryptographic API standards.

- **Roaming download of the ArcotID PKI**

  The ArcotID Client has the ability to download an ArcotID. Depending on which client is used, the ArcotID PKI can be temporarily downloaded into memory, saved on the user's hard disk, or saved on a USB storage device.

- **Signing a PDF by using a Roaming ID from CA SignFort in Adobe Reader**

  A subset of ArcotID PKI Client is embedded in Adobe Acrobat 8 (and higher) as well as Adobe Reader 8 (and higher). This functionality enables the use of an ArcotID PKI for two-factor authentication when digitally signing a PDF file by using a roaming digital ID that is stored in the CA SignFort server product.

ArcotID PKI Client Types

To support a wide variety of application environments, the ArcotID PKI Client is available in a variety of deployment types. These include:

- Flash Client (see page 11)
- Unsigned Java Applet (see page 11)
- Signed Java Applet (see page 11)
- Native Client for Windows and Mac OS X (see page 12)
- Embedded Client in Adobe Reader and Adobe Acrobat (see page 12)

Flash Client

The Flash client is an implementation of the ArcotID PKI Client that can run in a Web browser that has the Adobe Flash Player (version 9 or higher). The Flash client can be used only for strong authentication to Web applications or SSL VPNs. An ArcotID PKI downloaded by using the ArcotID PKI Flash client can be stored persistently. However, they cannot be accessed by other ArcotID PKI client types.

Unsigned Java Applet

The unsigned Java applet is an implementation of the ArcotID PKI Client that can run in any Web browser that contains a Java Virtual Machine (JVM) from Sun Microsystems or Microsoft. ArcotID PKIs downloaded by using the unsigned Java applet cannot be stored persistently. ArcotID PKIs can only be downloaded temporarily into memory for the duration of the browser session.

Signed Java Applet

The signed Java applet is an implementation of the ArcotID Client that can run in any Web browser that contains a Java Virtual Machine (JVM) from Sun Microsystems or Microsoft. When using the CA signed Java applet, the user will be presented a security message that requires the user to accept the signed applet before it is invoked. ArcotIDs downloaded by using the signed Java applet can be stored persistently. ArcotIDs downloaded by using the signed Java applet can also be accessed by the Native client for Windows.

A pop-up window is displayed when the signed Java applet is invoked for the first time. Click **Yes** on the window to continue using Java applet client.

Native Client for Windows and Mac OS X

The native client for windows is an install package that includes the Arcot browser plug-in, Arcot Cryptographic Service Provider (CSP), and Arcot PKCS#11 module. The native client offers more functionality that the other clients. When installing the native client, the user is presented with a security message that requires the user to accept the installation of the client software.

Native client is available on Mac OS X platform in the form of Netscape Plug-ins for Mozilla Firefox and Apple Safari browsers.

Embedded Client in Adobe Reader and Adobe Acrobat

ArcotID PKI Client functionality is embedded in the shipping versions of Acrobat 8 (and higher) and Adobe Reader 8 (and higher). This functionality enables ArcotID PKIs to be used for authentication to digitally sign PDF files by using a Roaming Digital ID. This functionality is dependent on another CA product, SignFort (available separately). ArcotID PKIs downloaded for use with SignFort are fully compatible with ArcotID PKI Native Client and Signed Java applet, and can be used for other non-Adobe ArcotID PKI client functionality.

Summary of Capabilities

The following table provides the quick overview of various client's functionality:

		Flash Client	**Unsigned Java Applet**	**Signed Java Applet**	**Javascript Client**	**Native Windows**	**Native Mac OS X**	**Embedded in Adobe Reader**
ArcotID PKI Roaming Download		+	+	+	+	+	+	+
ArcotID PKI Saved to Desktop		+		+	+	+	+	+
ArcotID PKI Saved to USB Drive						+		*
Authentication	Web	+	+	+	+	+	+	
	VPN	*	*	*	*	+		

	Flash Client	**Unsigned Java Applet**	**Signed Java Applet**	**Javascript Client**	**Native Windows**	**Native Mac OS X**	**Embedded in Adobe Reader**
Digitally Sign Web Forms	+	+	+	+	+		
Digital Signing and Encryption CSP (MS office) and PKCS#11 applications (PDF)					+		
Digital Signing with Roaming IDs							+
Device Locked ArcotID PKI			+	+	+		

+ Stored in flash secure object store, not available to copy to USB, floppy, CD, or other storage devices.

* Can use an ArcotID PKI stored on a USB drive, but cannot save to USB.

* Only for SSL VPNs through a Web browser.

Chapter 2: What's New

This chapter describes the new features in the ArcotID PKI Client.

New Features in 6.0.4.3

This section provides a quick glance at the new features and enhancements provided in ArcotID PKI Javascript Client release 6.0.4.3:

Note: The ArcotID PKI Javascript Client is now at release 6.0.4.3. No changes have been made in any of other clients.

- **Support for the AuthMinder Plugin Store**

  Support for the AuthMinder Plugin store has been introduced for storing ArcotID PKI credentials. This is in addition to the existing support for storing ArcotID PKI credentials in HTML5 local storage and the cookie store.

- **Support for Device Locking by Using the AuthMinder Plugin Store**

  Note: In this document, the terms device and machine refer to a mobile device or computer that is running an operating system supported by the ArcotID PKI.

- **Support for Device Authentication Using the ArcotID PKI Credentials**

  The ArcotID PKI can now be stored in a central location on a device, and it can be used to authenticate that device. An ArcotID PKI that is stored in the device store is automatically shared between all users of that device. This feature is also referred to as the Shared ArcotID PKI feature in this document. The ImportArcotID() method has been enhanced to accommodate this feature.

  Note: The Shared ArcotID feature requires that CA AuthMinder Plugin 2.1 or later must be installed on the end user's device.

New Features in 6.2

This section provides a quick glance at the new features and enhancements provided in 6.2 for ArcotID PKI Native Client:

- **Support for 64-Bit Microsoft Windows Operating System**

  ArcotID PKI Native Client now supports 64-bit Windows operating systems.

New Features in 6.1

This section provides a quick glance at the new features and enhancements provided in 6.1 for ArcotID PKI Native Client:

- **Support for Offline Access**

  The user's Open PKI keys and certificates that are stored in the ArcotID PKI can now be used offline. To enable this feature, the ArcotID PKI Native Client supports an utility called Arcot Offline Tool. Refer to chapter, "ArcotID (see page 89)PKI As a Software Smartcard" (see page 89) for more information on this feature.

- **Support for New Tool**

  To support the offline access feature, the ArcotID PKI Native Client now provides a new utility called Arcot Offline Tool. This tool is used to set offline password that is used for offline access, synchronize ArcotID PKI access, and reset or change offline password. Refer to "Working With Arcot Offline Tool" (see page 104) section for more information on the tool usage.

  The Arcot Offline Tool also provides an Online help that explains the tool usage.

- **Support for New Functions**

  To support the offline access feature, "CreateOfflineKeyBag()" (see page 49) and "UpdateOfflineKeyBag()" (see page 76) functions have been introduced.

- **Support for Password Cache**

  The ArcotID PKI online and offline passwords are now cached. The cached passwords can be used for a configured period or configured number of times. See "Password Caching" (see page 95) for more information on this.

New Features in 6.0.2.4

This section provides a quick glance at the new features and enhancements provided in 6.0.2.4 for ArcotID PKI Native Client:

- Internationalization (i18n) Support

  ArcotID PKI Native Client ATM Graphic User Interface (GUI) text can now be localized to different languages. See "ArcotID (see page 24)PKI Authentication" (see page 24) for more information.

- Support for Custom Storage Location

  You can now download the ArcotID PKI to a location of your choice. To enable this feature, you have to set the ArcotIDLocation parameter in the WebClient.ini file to the desired location.

  By default, ArcotID PKIs are downloaded to the %APPDATA%\arcot\ids folder.

  Note: This release also supports ArcotID PKIs that are available in the legacy default folder, which is %USERPROFILE%\My Documents\ArcotIDs.

New Features in 6.0.2

This release is mainly focused on enabling the user to login using the same ArcotID PKI with different aliases for the USERNAME. To support this feature, following new APIs are added:

- IsArcotIDAvailableEx() (see page 69)
- SignChallengeEx2() (see page 75)
- RemoveArcotIDEx() (see page 77)

New Features in 6.0.1

This section provides a quick glance at the new features and enhancements provided in 6.0.1 for ArcotID PKI Signed Java Applet (see page 11) client:

- **Device Locking**

  Note: Previously, this feature was available only for Native client.

  This feature enables an ArcotID PKI to be locked to one system, such that the ArcotID PKI is not usable if it is copied to another system.

  See section, "Machine PIN for Signed Java Applet Client" (see page 28) for more information.

- **Support for Linux Platforms**

  The following flavors of Linux are now supported:

  - Ubuntu 9.10
  - SUSE 11.2
  - Fedora 12

  The default ArcotID PKI storage location on these Linux distributions is same as Mac OS, which is:
  `$HOME/.arcot/ids`

- **Methods Currently Not Supported**

  The following methods are currently *not supported* for the **Signed Java Applet** client:

  - SignChallenge()
  - RegisterCSPCertificates()
  - SignChallengeNonBlocking()

  See chapter, "The ArcotID (see page 45)PKI Client API Reference" (see page 45) for more information.

New Features in 6.0

This section provides a quick glance at the new features and enhancements in 6.0:

- **Support for New Platforms**
  - This version extends the support of ArcotID PKI Client plug-in on Mozilla Firefox and Apple Safari browsers on Mac OS X platform. Refer to "Deploying on Mac OS X" (see page 83) for more information on the installation procedure.
  - This version extends the support of ArcotID PKI Client plug-in on Mozilla Firefox on Windows operating system.
- **API Changes**

  The following APIs are changed in this release to pass the organization name as an additional input parameter:
  - SignChallengeEx()
  - RemoveArcotID()
- **ArcotID PKI File Name Changes**

  ArcotID PKI files were previously saved as *username_domainname*.aid. The file format is now changed. The new file format is derived by hex-encoding the hash of <*username*><*organization*><*domain*> appended together. For example: `97A6D55BE5375DEFDA0CC825302E2E868234B438.aid`
- **ArcotID PKI Default Storage Location**

  On Windows: The new default location for storing an ArcotID PKI on Windows is %USERPROFILE%\My Documents\ArcotIDs.
  - For Windows XP users, the ArcotID PKIs will be stored at: `C:\Document and Settings\<username>\My Documents\ArcotIDs`
  - For Windows Vista users, the ArcotID PKIs will be stored at: `C:\Document and Settings\<username>\Documents\ArcotIDs`

    Note: Any ArcotID PKI stored in the legacy directory %APPDATA%\arcot\ids will continue to be recognized.

  On Mac OS X: The new default location for storing an ArcotID PKI on Mac OS X is $HOME/.arcot/ids directory.

Chapter 3: Supported Operating Environments

For information about supported operating system versions, see the Platform Support Matrix.

Chapter 4: ArcotID PKI Client Features

This chapter discusses the following ArcotID PKI Client features in detail:

- ArcotID PKI Storage (see page 21)
- ArcotID PKI Authentication (see page 24)
- JavaScript APIs (see page 25)
- DNS Restrictions (see page 25)
- Device Locking (see page 26)
- Client Logging (see page 28)
- Customizing ArcotID PKI Client Behavior (see page 33)
- Precedence Logic (see page 38)
- Partial Hash (see page 39)

ArcotID PKI Storage

The download of an ArcotID PKI can be done either for the current session or permanently. This is identified by the storage medium that will be selected while downloading the ArcotID PKI. This section lists the storage medium supported by ArcotID PKI Clients.

- ArcotID PKI Native Client (see page 21)
- Java Applet (see page 22)
- Flash Client (see page 23)
- JavaScript Client (see page 23)

ArcotID PKI Native Client

The ArcotID PKI Native Client supports three different locations to save the ArcotID PKI. You can choose the storage location during the ArcotID PKI download. The following are the storage locations supported by the Native Client:

- Storing on the Hard Disk (see page 22)
- Storing in the Memory (see page 22)
- Storing in the Universal Serial Bus (USB) (see page 22)

Storing on the Hard Disk

The ArcotID PKI is directly downloaded to the user’s hard disk. By default, the ArcotID PKI is downloaded to the %APPDATA%\arcot\ids folder.

ArcotID PKI Native Client also enables you to store ArcotID PKIs in a custom location. To download and store ArcotID PKIs in a custom location:

1. Navigate to the <%SYSTEMDRIVE%>:\Program Files\Common Files\Arcot Shared\conf folder.
2. Open the WebClient.ini file in a text editor.
3. Set the ArcotIDLocation parameter to the location where you want to download the ArcotID PKI.
4. Save and close the WebClient.ini file.

Note: If there is any error while storing the ArcotID PKI in a custom storage location, then the OnErrorStoreInDefaultLocation parameter is checked. If the value of this parameter is set to TRUE, then the ArcotID PKI is stored in the default location. Otherwise, an error is returned.

Storing in the Memory

The ArcotID PKI is downloaded to memory, and this is available only for the current session. The user has to re-download the ArcotID PKI if required for the successive sessions.

Storing in the Universal Serial Bus (USB)

The ArcotID PKI is downloaded to the USB that is plugged to the system. If the ArcotID PKI Client detects more than one device, then a pop-up window appears, which lists all the available external storage medium. The user has to select a device from the list.

If the machine is not connected to any external device, then the user will be displayed with appropriate message.

Java Applet

ArcotID PKI Java Applet client enables you to download the ArcotID PKI to your file system or store it in the browser memory.

Refer to "Storing on the Hard Disk" (see page 22) and "Storing in the Memory" (see page 22) for more information on the storage methods supported by Java Applet.

Flash Client

The Flash client stores the ArcotID in the %APPDATA%\Macromedia\Flash Player\#SharedObjects\<*unique_number*>\<*domain_name*> folder. The ArcotID folder name is same as the domain name from where the ArcotID Flash client movie is downloaded.

JavaScript Client

JavaScript ArcotID PKI Client supports the following types of storage medium:

- In Memory Storage (see page 23)
- Permanent Browser Storage (see page 23)

In Memory Storage

The ArcotID PKI is downloaded to memory, and this is available only for the current session. The user has to re-download the ArcotID PKI if required for successive sessions.

Permanent Browser Storage

The ArcotID PKI JavaScript Client supports HTML5 standard to store ArcotID PKIs in the Web browser. According to this standard, the ArcotID PKI is stored permanently in the Web browser and is available across sessions.

Note: If an older version of the Web browser that does not support HTML5 standard is used to download the ArcotID PKI, then the ArcotID PKI is stored as a permanent browser cookie. In such cases, there is a restriction on cookie size, therefore a user can store *only* one ArcotID PKI on their system.

ArcotID PKI Authentication

The ArcotID PKI authentication process begins when a user connects to a Web page that requires authentication. The ArcotID PKI authentication process is done in the following steps:

- Send Appropriate Challenge
- Generate Signature
- Verify Response
- Verify Authentication Token

The ArcotID PKI Client collects the user password for ArcotID PKI to access the application or the resource that is protected by CA AuthMinder. An ATM GUI is provided to collect the ArcotID PKI password. The password can either be entered by using keyboard or the pin-pad. See "Issuer Preferences" (see page 33) for more information on the PIN input method.

The text of the ATM GUI can be localized to different languages. ArcotID PKI Native Client currently supports the following international languages:

- American English
- German
- Danish
- Swedish
- Dutch
- Portuguese
- Spanish
- French
- Japanese
- Russian
- Italian
- Turkish

JavaScript APIs

ArcotID PKI client provides a rich set of APIs for customizing the ArcotID PKI display and behavior. See chapter, "The ArcotID PKI Client API Reference" (see page 45) for more information on Java APIs.

Note that the following methods are currently *not supported* for the **Signed Java Applet** client:

- SignChallenge
- RegisterCSPCertificates
- SignChallengeNonBlocking

DNS Restrictions

When an ArcotID PKI is downloaded, the primary DNS domain where it was downloaded from is recorded in the ArcotID PKI. The user will not be permitted to use that ArcotID PKI in any other primary DNS domain.

For example, if an ArcotID PKI is downloaded from A.ARCOT.COM, then it can be used for authentication at B.ARCOT.COM and C.ARCOT.COM, or any server in the ARCOT.COM domain. However, it cannot be used at Web servers in other primary domains, for example, A.ARCOTSYSTEMS.COM.

Device Locking

The Device Locking feature enables an ArcotID PKI to be *locked* to a specific machine, so that the ArcotID PKI is not usable if it is copied to another machine.

The feature works by camouflaging (protecting) an ArcotID PKI using a password made of two components.

1. The password selected by the user when the ArcotID PKI is issued.
2. A new Machine PIN, which is derived from unique machine-specific information derived from the hardware characteristics of the client machine.

When device locking is enabled, the ArcotID PKI is cryptographically camouflaged twice, once with the user password and once with the Machine PIN.

The device locking is done at the time when an ArcotID PKI is downloaded to the user's machine. After an ArcotID PKI is locked to the user's machine, it is not usable if you copy it to another machine.

Typically when device locking is enabled, the server will not enable Roaming of the ArcotID PKI, so that the user will not be able to download the ArcotID PKI to another machine. To enable both roaming on the server and device locking, the ArcotID PKI is device locked separately to each machine on which it is downloaded.

Machine PIN for Native Client

The Machine PIN is derived from the computer-specific information. Not all the information type is present on all the computers. For example, the motherboard serial number is not present on all motherboards. The ArcotID PKI Client uses such machine characteristics available on the computer.

The following are the set of categories that are considered for generating the Machine PIN:

- mem

  Physical memory size.

- vol

  Boot partition volume ID.

- bios

  BIOS information such as, serial number and manufacturer name. This information is not always present.

- mac

  MAC addresses of all fixed Network Interface Cards (NIC). NICs that are not fixed, for example, PCMCIA, USB, docking station and NIC that is virtual, for example, VPN adapters are skipped. All other NICs are included. Adding or removing a NIC will cause Machine ID to change.

- moth

  Motherboard information such as, the serial number and manufacturer name. This information is not always present.

- hd

  Hard Disk (HD) information such as, model number and manufacturer's name. This is not HD serial number and therefore it would be same for all identical HDs. The removable HDs such as external USB, memory cards, are not included. Only the fixed HDs are included and therefore adding or removing a HD will cause Machine ID to change.

- proc

  Main Central Processing Unit (CPU) information such as, model and clock speed. For multiprocessor machine the information from all CPUs is included. This is not the CPU serial number.

- encl

  Some computers, such as those manufactured by Dell, have their service tag encoded in the enclosure information.

Machine PIN for Signed Java Applet Client

The **Signed Java Applet** client currently uses vol and lhname for generating the Machine PIN. It tries to obtain the information for both attributes to construct the Machine PIN. If that fails, then it uses the machine characteristics available on the computer. If none of the attributes are available, then download of ArcotID PKI will fail.

- vol

  The volume ID of the Boot partition.

- lhname

  The local host name of the system to which the ArcotID PKI will be downloaded.

  Note: To specify both categories for device locking, set the devlock_type parameter to vol_lhname.

Client Logging

ArcotID PKI Native Client supports logging for all the user actions. It also supports file logging for the server and other component logs. Logging can be done in multiple steps with a fine-grain configuration during startup and system management.

This section covers the following topics:

- Enabling Logging (see page 29)
- Supported Log Levels (see page 30)
- Log File Configurations (see page 32)

Enabling Logging

Perform the following steps to enable logging:

1. Copy the ArcotLog2FileSC.dll file to the <%SYSTEMDRIVE%>:\system32\ directory.

   Note: ArcotLog2FileSC.dll is not a part of the standard client package. It is only required for advanced troubleshooting purposes and is not required for most users. To obtain this DLL, please contact Arcot Technical Support.

2. Navigate to the following directory:
   `<%SYSTEMDRIVE%>:\Program Files\Common Files\Arcot Shared\conf`

3. Open WebClient.ini file in the text editor.

4. In the arcot/WebClient section, set the LogSupported parameter to the number of loggers to send messages to.

   By default, this parameter is set to 1.

5. Create the [arcot/WebClient/LogLibrary<*n*>] sections based on the number to which the LogSupported parameter is set to. For example, if LogSupported is set to 2, then you must have [arcot/WebClient/LogLibrary1] and [arcot/WebClient/LogLibrary2] entries.

   The [arcot/WebClient/LogLibrary<*n*>] section must have the following entries.

   - DLLName

     The name and path of the shared library that enables logging. If you do not provide the full path, then the PATH variable is used to search the file.

     Note: The file name suffix is added automatically, therefore you need *not* provide the suffix.

   - HandleType

     Defines the type of messages that must be logged. See "Supported Log Types" (see page 30) for more information.

   - HandleLevel

     Defines the level of information detail stored in the log file. See "Supported Log Levels" (see page 30) for more information.

   - EntryPoint

     The function that is used to get a handle to the logging object. This is fixed for a log handler library.

   - ParamSupported

     Indicates the number of objects to be passed to the log handler. See "Log File Configurations" (see page 32) for more information.

   - Param<n>

     Log name-value pair declaration based on the value of the ParamSupported parameter. See "Log File Configurations" (see page 32) for more information.

6. Save and close the file.

Supported Log Types

The HandleType parameter in the WebClient.ini file describes the type of log messages that have to be logged. The following table lists the log types that the HandleType parameter supports:

Log Type	**Description**
ALL	All the requests processed by the ArcotID PKI Client are logged.
APPLICATION	The information specific to the ArcotID PKI Native Client. For example, if CSP or PKCS#11 APIs are not implemented, if unsupported PKCS data format is parsed, or if there is an error in the other important steps in application functionality.
SYSTEM	This type is used for system events. For example, failure in writing to the memory. Logs of this type are currently *not* generated by the ArcotID PKI Native Client.
AUDIT	The type is used to record the user transactions. Logs of this type are currently *not* generated by the ArcotID PKI Native Client.
STATUS	The requests that are related to ArcotID PKI Client status checks are logged.
TRACE	ArcotID PKI Client also provides trace logging, which contains the flow details. The trace logs, which are logged in the configured log file help during troubleshooting. The entries for the trace messages start with TRACE:.

Supported Log Levels

A *handle level* (or *severity level*) enables you to specify the level of detail of the information stored in the log file. This also enables you to control the rate at which the log file will grow.

You need to use the HandleLevel parameter in the WebClient.ini file to set the log levels in the log file. The following table describes the values that the HandleLevel parameter supports:

Log Level		**Description**
1	debug	Use this log level for logging detailed information for debugging purposes. This might include process tracing and changes in Server states.

Log Level		**Description**
2	info	Use this log level for capturing information on run-time events. In other words, this information highlights the progress of the application, which might include: ■ Changes in the ArcotID PKI file ■ Import new certificates to the Windows certificate store.
3	notice	Use this log level for checking the ArcotID PKI Client status and version.
4	warning	Use this log level for undesirable run-time exceptions, potentially harmful situations, and recoverable problems that are not yet FATAL.
5	error	Use this log level get information about failures in functionality that occur due to the data in unsupported format. For example, parsing corrupt ArcotID PKI files, or corrupt key and certificate data.
6	alert	Logs of this level are currently *not* generated by the ArcotID PKI Native Client.
7	fatal	Use this log level for serious, non-recoverable errors that can cause abrupt termination of the AuthMinder service.

Note: When you specify a log level, messages from all other levels of higher significance are reported as well. For example if the HandleLevel is specified as 3, then messages with log levels of debug, info, and notice level are also captured.

Log File Configurations

To set the ArcotID PKI Client log file configurations, you need to use the [arcot/WebClient/LogLibrary<*n*>] section of the WebClient.ini file. This file is available at the following location:
`<%SYSTEMDRIVE%>:\Program Files\Common Files\Arcot Shared\conf`

The log parameters are defined as name-value pairs. Set the ParamSupported parameter to the number of the name-value pairs that you plan to define. By default, the value of ParamSupported is set to 4, which means four logging parameters can be configured.

The following table lists the default logging parameters and their values that the ArcotID PKI Client supports:

Parameter	**Name**	**Description**
Param1	LOG_FILE_NAME	Specify the log file name and the location where the ArcotID PKI Client logs must be written to. By default, the ArcotID PKI Client log file name is ArcotClientLog.txt and is created in the *<%SYSTEM_DRIVE%>*\windows\system32\ folder.
Param2	LOG_FILE_ROLLOVER_INTERVAL	Specify the duration after which the log file contents must be moved to the backup file. Supported values are: ■ HOURLY ■ DAILY ■ WEEKLY ■ MONTHLY By default, the rollover interval is set to DAILY, which indicates that the rollover happens when the first messaged is logged after the midnight. **Note:** If you do not set any value for this parameter, then the log rollover happens when the log file size is equal to MAX_LOG_FILE_SIZE.
Param3	MAX_LOG_FILE_SIZE	Specify the maximum size of the log file. By default, the maximum log file size is 10 MB.
Param4	BACKUP_LOG_FILE_LOCATION	Specify the location where the backup log file must be saved. By default, the ArcotID PKI Client backup log file is created in the *<%SYSTEM_DRIVE%>*\windows\system32\ folder.

Parameter	Name	Description
Param5 **Note:** By default, this parameter is *not* enabled.	LOG_LINE_FORMAT	Specify the attributes that should be logged on each log line. Following attributes are supported: ■ Time stamp (TS1L) The time and date when the entry was logged, translated to the local time zone. The format of logging this information is as follows: day month date HH:MM:SS.mis year. Here, mis represents milliseconds. ■ Log level (SEV) The severity level of the logged entry. See "Supported Log Levels" (see page 30) for more information. ■ Process ID (PID) The ID of the process that logged the entry. ■ Thread ID (TID) The ID of the thread that logged the entry. ■ Message ID (MID) The unique identifier of the message. ■ Message (MSG) The log message. ■ Logging ID (LID) The ID of the logger. **Note**: This attribute is currently *not* supported.

Customizing ArcotID PKI Client Behavior

The ArcotID PKI Client provides the following three tiers of preferences for customizing the behavior of the ArcotID PKI Client application:

- Issuer Preferences (see page 33)
- Application Preferences (see page 37)
- User Preferences (see page 37)

Issuer Preferences

Issuer preferences are name-value pairs stored in an ArcotID PKI. These preferences are also referred to as ArcotID PKI attributes. The supported attributes in the ArcotID PKI Client include:

url_main

URL of the ArcotID PKI issuer (for informational purposes only.)

Banner

Binary data of the logo image present on ArcotID PKI.

url_help

URL where the user can look for help related to this ArcotID PKI.

input_method

Determines the way how the ArcotID PKI password can be received by the ATM-style GUI. Possible values include:

- mouse

  Input can only be entered by using the mouse. Pin-pad is visible but password edit box is disabled.
- keyboard

  Input can only be entered by using the keyboard. Pin-pad is hidden and password edit box is enabled.
- dual

  Both mouse and keyboard can be used for entering the input. Pin-pad is visible and password edit box is enabled.
- not set

  Required to be set. If not set, an undefined error appears.

userID

User ID of the ArcotID PKI. The User ID and the card name of the ArcotID PKI are stored in an extension in the Arcot certificate.

Org

Organization name of ArcotID PKI.

Domain

Domain name of DNS domain where the ArcotID PKI was downloaded. This attribute is set by the client software at the time when the ArcotID is downloaded, for example, arcot.com.

Key Store

The binary value of the *key bag*. Additional application keys protected by using the Key Authority features of AuthMinder.

KeyFortURL

The URL of the AuthMinder Key Authority (optional).

KeyFortUseWinInet

Determines whether or not native WinInet or WinHTTP, or both the libraries should be used for HTTP communication for Key Authority.

KeyFortSecret

It can be used to have an additional secret to protect the keybag, beyond the normal process.

scrstyle

Determines how often the pin-pad should be scrambled in the ATM GUI. This is ignored if input_method is set to keyboard. Possible values include:

- never

  Pin-pad is never scrambled.

- once

  Scramble the pin-pad only once, when pin-pad is initially displayed.

- always

  Scramble the pin-pad every time a pin-pad key is clicked or pressed.

- not set

  The default value *once* will be set.

scrorder

Determines the manner in which the pin-pad is scrambled. This attribute is ignored if input_method is set to keyboard or if scrstyle is set to never. Possible values include:

- random

  Scrambling is done randomly. For example, "7253081496".

- sequential

  Scrambling is sequential. For example, "4567890123". The sequences "9012345678", "0123456789" are not allowed because the values are not scrambled completely.

- not set

  The default value random is set.

storage_type

Configures allowable storage medium for the ArcotID. Possible values include hd_usb_memory, or any combination of hd, usb, and memory delimited by the an *underscore* character. The default behavior is any. See "ArcotID Storage" (see page 21) for more information.

devlock_required

Configures whether the device locking feature is required for the ArcotID PKI. Possible values are yes and no. The default value for the devlock_required attribute (if it is not present or holds an invalid setting) is no.

Note: If the value is set to no, then the device locking can still be enabled by JavaScript using SetAttribute (DeviceLocking) because of the higher precedence.

devlock_type

A sequence of parameters delimited by underscores. For example, moth_bios_hd. The value all means all parameters are included for device locking. If no valid attribute is provided, then the default value all is used (assuming device locking is required or enabled by Application preferences).

Note: A value of no does not rule out the use of device locking if the application JavaScript sets a preference using SetAttribute(DeviceLocking).

Application Preferences

Application preference enables a Web application to customize the behavior of the ArcotID PKI Client at run-time within the application. Application preferences are set using the SetAttribute() JavaScript API function. See "SetAttribute()" (see page 55) for more information about the API and the supported attributes.

User Preferences

The ArcotID PKI native plugin client currently supports two user preferences. The user preferences are supported only in the native client.

Set default credential

In the **Arcot Digital Identity Plugin Options** dialog, the user can select the default credential that is selected in the ATM GUI. This feature is for user convenience only.

Show all ArcotID PKIs

In the Arcot Digital Identity Plugin Options dialog of the native plugin, the user can select the Show all ArcotID PKIs option to display all the ArcotID PKIs, even if they belong to different domain or filtering options.

Precedence Logic

The ArcotID PKI Client behavior is determined by the settings that can be configured in four main ways:

- By the client itself
- Inside an ArcotID PKI
- From a Web page through the JavaScript API
- Through choices made by the user at runtime

The precedence logic feature formalizes the precedence rules of how the run-time value of each setting is determined from the four sources of configuration information.

Note: The precedence rules themselves cannot be customized.

In general, the highest priority preferences are the issuer preferences. The issuer preferences specify the allowable behaviors for the ArcotID PKI. If an issuer preference is not set, then the preference is for the application preferences and then the user preferences.

The precedence rules are described in the following table:

Features	**Client Configuration**	**Issuer Preferences**	**Application Preferences**	**User Preferences**
Storage type for ArcotID PKI storage during download	Default values	Allowable values	Refinement of allowable or default values. Decider if no settings on ArcotID PKI	Refinement of allowable values.
Location of ArcotID PKI (during usage)	Default value		Overrides default value using credential filtering attributes	
Temporary or Permanent Download during Roaming	Permanent allowed only on some clients	Allowable values	Refinement of allowable or default values. Decider if no settings on ArcotID PKI	Refinement of allowable values

Features	**Client Configuration**	**Issuer Preferences**	**Application Preferences**	**User Preferences**
Device Locking	Enabled on some clients only	Allowable mechanisms for device locking	Refinement of allowable or default values. Decider if no settings on ArcotID PKI	N/A*
ArcotID PKI Image displayed (applicable only when ATM GUI is used)	Default value	Overrides default value when present	N/A*	N/A*
PIN PAD scramble method	Default value	Overrides default value	Overrides ArcotID PKI values, but only within the set of possible values allowed inside the ArcotID PKI	

*The feature is not applicable.

Partial Hash

Within the ASN.1 structure of an ArcotID PKI (not in the ArcotID PKI attributes), an ArcotID PKI may store a partial hash of the password. This is an optional feature and it is controlled by the server that creates the ArcotID PKI.

The presence of the partial hash enables a portion of the invalid password to be tested on the client-side. This prevents user lockouts due to typos. However, systematic attempts like brute-force attack on the password will still be prevented.

The size of the partial hash is also configurable when the partial hash is stored in the ASN.1 structure. Configuring the size of the partial hash can help balance the trade-off between convenience (less user lockouts) and security (preventing attackers who are attempting to guess the password.)

Note: The partial hash functionality may be deprecated in the future release.

Chapter 5: Invoking the ArcotID PKI Client

This chapter describes invocation process for different ArcotID PKI Client from a Web page.

Invoking ArcotID PKI Client on Web Page

This section describes the process of invoking ArcotID PKI Client on a Web page. Refer to the appendix, "Source Code Samples" (see page 123) appendix for the working sample code that is implemented by using these instructions.

To invoke the ArcotID Client from a Web page:

1. Include arcotclient.js in your page using the SCRIPT tag as follows:
   ```
   <SCRIPT type="text/javascript"
   src="arcotclient.js"></SCRIPT>
   ```
2. Create a DIV tag inside the body of your page.

   This identifies where dynamically generated code will be inserted. Depending on the type of client invoked (Native, Java), different code may be generated in JavaScript. The code that is generated will be automatically inserted at the location of this DIV tag.

   The DIV tag needs to include a unique identifier such as ArcotIDClient. This unique identifier is used in the next step.

   The DIV tag is coded as follows:
   ```
   <div id="ArcotIDClient"></div>
   ```

3. Add JavaScript initialization method. Create a JavaScript method that initializes the ArcotID PKI Client when the page loads. For example:

```
var arcotClient;
function initClient() {
arcotClient = new ArcotClient();
arcotClient.setAttribute("clientBaseURL", "client");
arcotClient.write("ArcotIDClient");
   }
```

   In the preceding example, clientBaseURL is an optional attribute. The full list of optional client invocation attributes are listed in the following table:

Attribute	**Possible Values**	**Description**	**Default Value**
clientType	ActiveX Applet	Sets which client should be used. Can be invoked several times to specify an order of preference for which client should be used. For example, if it is called first with Flash and then with Applet, then when write() is invoked, the library will first try to use the Flash version and if a Flash player is not available (or not the right version of Flash), it will instead use the Applet.	The default behavior is that the library detects the most appropriate client for the user's browser and OS combination.
clientBaseURL	Any URL (relative or absolute). If absolute, it must start with http:// or https://	Sets the URL of the directory where the different ArcotID PKI Client package files (.cab, .jar, .swf, .js) can be found.	The default value is "", which means the client files are expected to be found at the same directory level as the invoking Web page.

Attribute	**Possible Values**	**Description**	**Default Value**
clientReadyCallback	The name of a JavaScript method on the current page.	This defines a JavaScript callback method that is invoked when the ArcotID PKI Client is fully initialized and ready to process API calls such as SignChallenge() and GetGlobalAttribute(). This callback can be used to ensure login buttons are disabled on a Web page until after the client loads. This is especially useful for the Java applet because the JVM can take time to initialize during the first invocation of the applet. It is strongly recommended to use this callback to avoid errors that result if ArcotID PKI Client API functions are invoked before the client is initialized and ready.	None
flashInstallURL	URL (relative or absolute). If absolute, it must start with http:// or https://	URL where the user is redirected if Flash is not installed in the user's browser.	*http://www.macromedia.com/go/getflashplayer*
flashUpdateURL	URL (relative or absolute). If absolute, it must start with http:// or https://	URL the user is redirected to if Flash is available in the user's browser but is not the required version	*http://www.adobe.com/products/flash/about/*
javaInstallURL	URL (relative or absolute). If absolute, it must start with http:// or https://	URL the user is redirected to if Java is not available in the user's browser	*http://www.java.com/en/download/*
signedApplet	true or false	Whether the signed applet should be used instead of the unsigned applet.	true

Attribute	Possible Values	Description	Default Value
ActiveXMinVersion	A version string in the following format: 5,0,0,0	Sets the minimum version of the ArcotID PKI ActiveX plugin that is required by the Web page. If the user has an older version of the ActiveX plugin, then the user will be prompted to upgrade to the newest version.	5,0,0,0

4. Add onload event.

   Add a call to the initClient method (created in Step 3) to the onload event of the body tag as follows:
   ```
    <body onload="initClient();">
   ```
   The onload event only happens after all the resources of the pages have been loaded. Pages that have large images might take more time to load, causing the client to be instantiated only after that extra delay. An alternative to speed-up the loading process is to instead use the ready event (which is triggered as soon as the page structure is available). Because of browser incompatibilities, Arcot suggests you to use a third-party package such as jquery. Using jquery, you can replace the onload event with the following JavaScript lines:
   ```
   $(document).ready(function(){
   arcotClientInit();
   }
   ```

Chapter 6: ArcotID PKI Client API Reference

This chapter contains reference material for the client-side development with AuthMinder. All of the client-side controls for AuthMinder are provided in the ArcotID PKI Client Javascript API. This API allows you to program the functionality of the client using various Web programming languages including Java and JavaScript.

ArcotID PKI Client Javascript APIs

The ArcotID PKI Client Javascript API controls all the client-side functions related to managing and authenticating ArcotID PKIs. These APIs are common for both ArcotID PKI Client browser plug-in and ArcotID PKI Client Applet. These include signing challenges and managing error messages. Before the API can be accessed by the client, the AuthMinder browser plug-in must be installed on the client system.

Methods	**Description**
AddCurrentCardToWallet() (see page 47)	Adds the current ArcotCard to the ArcotID PKI.
AttachCertToCurrentCard() (see page 48)	Attaches the new certificate to the ArcotCard.
CreateOfflineKeyBag() (see page 49)	Creates an offline key bag for the user and stores it in their ArcotID PKI.
GetErrorCode() (see page 50)	Returns an error code for the last unsuccessful call to the browser plug-in.
GetErrorMessage() (see page 53)	Returns a readable string that describes the last error encountered.
GetVersion() (see page 53)	Requests the version number of the currently loaded plug-in or applet.
GetVersionEx() (see page 74)	Gets the extended version information of the native client or the applet.
GetGlobalAttribute() (see page 54)	Requests a previously set plug-in attribute value.
SetAttribute() (see page 55)	Sets the value of predefined plug-in attributes and creates new plug-in attributes.
SignChallenge() (see page 65)	Signs a challenge previously obtained from the authentication server.
SignChallengeEx() (see page 66)	Signs a challenge previously obtained from the authentication server.

Methods	Description
SetCurrentCardByIndex() (see page 61)	Sets the current card to be the one with the given index in the current wallet.
SetCurrentWalletFromEncoding() (see page 63)	Sets the current wallet to be the one that is passed in as a string parameter.
ImportArcotID() (see page 67)	Downloads the ArcotID PKI.
IsArcotIDAvailable() (see page 69)	Checks if the ArcotID PKI of the user is present on the system.
IsArcotIDAvailableEx() (see page 69)	Checks if the ArcotID PKI of the user is present based on the lookup mode.
RemoveArcotID() (see page 71)	Removes the ArcotID PKI.
RegisterCSPCertificates() (see page 72)	Registers the certificates with Microsoft CAPI.
SignChallengeNonBlocking() (see page 73)	Enables to support the ATM GUI applet on Mac OS X. The API call receives callback functions for success and error handling.
RefreshArcotIDs() (see page 75)	Instructs the ArcotID PKI client to re-read the storage areas for ArcotID PKIs.
SignChallengeEx2() (see page 75)	Signs a challenge based on the lookup mode.
RemoveArcotIDEx() (see page 77)	Deletes the ArcotID PKI of the user based on the lookup mode.
UpdateOfflineKeyBag() (see page 76)	Updates the offline key bag of the user.

AddCurrentCardToWallet()

This method adds the current Arcot card to the named wallet stored in the client machine. If the mentioned file is not present, a new one is created.

Note: This is a deprecated function, use ImportArcotID() (see page 67) instead.

Syntax

```
boolean AddCurrentCardToWallet(WalletName)
```

Parameters

The following are the parameters of this method:

Parameter	Type	Description
WalletName	string	File name for the new wallet file (without the .wlt extension.)

Returns

If the method is successful, then it returns TRUE. If the method is unsuccessful, then it returns FALSE.

Example

```
var arcotClient = new ArcotClient();
var walletnameString = "GuestUser";

if (arcotClient.AddCurrentCardToWallet(walletnameString))
{
    document.write("<P>Card added to wallet successfully</P>");
}
else
{
    document.write("<P>Failed to add card to wallet</P>");
}
```

AttachCertToCurrentCard()

This method is used as part of the client-side key generation. AttachCertToCurrentCard() is used to attach the newly generated certificate to the card, where the keys were generated from GetErrorCode() (see page 50).

After this method, AddCurrentCardToWallet() (see page 64) needs to be invoked to add the new card to a wallet and store the wallet to the user's machine.

Syntax

```
string AttachCertToCurrentCard(certificateEncoding)
```

Parameters

The following are the parameters of this method:

Parameter	Type	Description
certificateEncoding	string	Base-64 encoded certificate you wish to attach to the current card.

Returns

If the method is successful, then it returns TRUE. If the method is unsuccessful, then it returns FALSE.

Example

```
var arcotClient = new ArcotClient();
var b64certString = "MIIFeQIBAAwHcGlucGFkMTCCBWcwggVjAgEBDAR
                     jYXJkoIHFMB8GCmCGSAGG+UYJAQEwEQQAAgEBAg
                     ILAgICAegCAgMYBIGhMIGeAgEAMA4GCmCGSAGG+
                     UYJAgAFAASBiDCBhQJBAM4mHRH5WMYDjDPUZD16...";

if (arcotClient.AttachCertToCurrentCard(b64certString))
{
    document.write("<P> AttachCertToCurrentCard succeeded</P>");
}
else
{
    document.write("<P> AttachCertToCurrentCard failed</P>");
}
```

CreateOfflineKeyBag()

This method is used to create an offline key bag and store it in the user's ArcotID PKI. The offline key bag contains a copy of user's certificates, which help users to access their protected data offline.

Syntax

```
boolean CreateOfflineKeyBag(walletName, orgName, onlinePIN,
offlinePIN)
```

Parameters

The following are the parameters of this method:

Parameter	**Type**	**Description**
walletName	string	Wallet name of an ArcotID PKI for which the offline key bag has to be created.
orgName	string	The organization name to which the user belongs to.
onlinePIN	string	The ArcotID PKI password that is used to authenticate to the Key Authority server and fetch the latest key bag.
offlinePIN	string	The password that the user needs to use to access their certificates offline.

Returns

If the method is successful, then it returns TRUE. If the method is unsuccessful, then it returns FALSE.

Example

```
var arcotClient = new ArcotClient();
var  userIDStr  = "johnsmith";
var orgName = "Acme Vendor";
var onlinePINStr = "12345^";
var offlinePINStr = "123456"
var response = arcotClient.CreateOfflineKeyBag(userIDStr, orgName, onlinePINStr,
offlinePINStr);
```

GetErrorCode()

Retrieves the error code for the last error encountered by the SDK.

Syntax

```
int GetErrorCode()
```

Returns

An error code.

Discussion

Possible values include:

Error Code	**Description**
ERR_NONE (0)	
ERR_BAD_PIN (1)	The user entered an invalid PIN.
ERR_GUI_CANCEL (2)	The user cancelled authentication before it completed.
ERR_MISSING_WALLET (3)	The specified Arcot card was not found.
ERR_MISSING_CARD (4)	The specified card was not found.
ERR_GUI_RENEW (5)	The user choose to renew instead of authentication.
ERR_BAD_WALLET (6)	The Arcot card is invalid.
ERR_INVALID_PUBLIC_KEY (7)	The public key you are attempting to use is invalid.
ERR_MISSING_SERVER_NAME (8)	The authentication server could not be found.
ERR_INITCONN (9)	The connection between the client and the application server failed to initialize.
ERR_CONNECT (10)	The client failed to connect to the authentication server.
ERR_CREATE_SYSTEM_ CONTEXT (11)	Could not create an Arcot system context.
ERR_CREATE_CONTEXT (12)	Could not create an Arcot context.
ERR_PROTOCOL (13)	An Arcot authentication protocol error.
ERR_AAPLIB (14)	An Arcot authentication library error.
ERR_AUTHENTICATE (15)	The authentication failed.

Error Code	**Description**
ERR_COOKIE (16)	The cookie placed in the browser failed to authenticate.
ERR_ILLEGAL_CHALLENGE (17)	The challenge sent to the authentication server failed.
ERR_UNKNOWN (18)	An unknown error occurred during the authentication process.
ERR_BAD_WALLET_FOLDER (19)	Error in the folder containing the wallet.
ERR_ILLEGAL_DOMAIN_ACCESS (20)	An error occurred while accessing the domain.
ERR_INVALID_CREDENTIAL_FILTER (21)	The user credential was invalid after the filtering result.
ERR_BAD_APPCTX (22)	The application context is invalid.
ERR_BAD_ALIAS (23)	The alias is invalid.
ERR_BAD_WALLET_OR_ALIAS (24)	The alias or the wallet name is invalid.
ERR_INVALID_STORAGE (30)	Storage medium mentioned is invalid.
ERR_STORAGE_UNAVAILABLE (31)	Storage mentioned is not available.
ERR_STORAGE_ERROR (32)	Error occurred while storing the ArcotID PKI.
ERR_NO_SUCH_WALLET_ERROR (33)	Mentioned wallet does not exist.

Example

```
<HTML>
<HEAD>
<TITLE>Authentication Failure</TITLE>
</HEAD>

<BODY>
<H2>Authentication Failure!</H2>

<SCRIPT LANGUAGE="JavaScript">

var arcotClient = new ArcotClient();
var errorcode = arcotClient.GetErrorCode();
if (errorcode == 1)
{
    document.write("<P>Invalid PIN</P>");
}
if (errorcode == 3)
{
    document.write("<P>Invalid User Name</P>");
}
```

GetErrorMessage()

Returns a text string that describes the last error encountered.

Syntax

```
string GetErrorMessage()
```

Returns

A readable string that describes the last error encountered.

Example

```
<HTML>
<HEAD>
<TITLE>Authentication Failure</TITLE>
</HEAD>

<BODY>
<H2>Authentication Failure!</H2>

<SCRIPT LANGUAGE="JavaScript">

var arcotClient = new ArcotClient();
var reason = arcotclient.GetErrorMessage();
document.write("<P>" + reason + "</P>");
```

GetVersion()

Gets the version number of the currently loaded Arcot plug-in or applet. This function can be used to check if the installed plug-in is latest or old.

Syntax

```
string GetVersion()
```

Returns

The version of the Arcot browser plug-in installed on the client system. For example, 4.6.0.0.

Example

```
<HTML>
<HEAD>

<SCRIPT LANGUAGE="JavaScript">

var arcotClient = new ArcotClient();
var version = arcotClient.GetVersion();
```

GetGlobalAttribute()

Retrieves attribute values from the plug-in, applet, or from a specific ArcotID PKI. Not all ArcotID PKI attributes can be retrieved by using this API.

Syntax

string GetGlobalAttribute(*attributeName*)

Parameters

The following are the parameters of this method:

Parameter	Type	Description
attributeName	string	The attribute you are requesting.

Returns

If successful, then it returns the Arcot card attribute you requested.

Discussion

Attributes allow you to set information that will persist in the plug-in till the browser session is closed. You can create your own attributes by using SetAttribute() or you can use special pre-defined attributes to obtain specific information about Arcot wallets and cards. Pre-defined attributes include:

Attribute	Description
walletn:count	Indicates the number of ArcotID PKIs present on the client.
walletn:*x:name*	Indicates the name of specific wallet.

Example

```
var arcotClient = new ArcotClient();
var i = 0;
var walletname;

// Get the number of wallets
var walletcount = arcotClient.GetGlobalAttribute("walletn.count");

// Print out the names of all the wallets
for (i = 0; i < walletcount; i++)
{
    walletname = arcotClient.GetGlobalAttribute("walletn:" + i +  ":name");
    document.write("<P>" + walletname + "</P>");
}
```

SetAttribute()

Sets an attribute that will persist in the plug-in or applet till the browser session is closed. This function is used to set the Application Preferences (see page 37).

Syntax

```
boolean SetAttribute(attributename, attributeValue)
```

Parameters

The following are the attributes that can be set by using this API function, all the attributes are of *string* type.

- WalletInMemory
- StorageType
- CredentialFilter
- AID_LOOKUP_MODE
- DeviceLocking
- ScrambleStyle
- ScrambleOrder

WalletInMemory

WalletInMemory is an attribute that instructs the client to download the ArcotID PKI to memory only if set to yes, or to disk (permanent) if set to no.

Note: If the old APIs are used, then use this attribute is used to store the wallet in memory or in the hard disk.

Note: This is a deprecated attribute, use StorageType instead.

StorageType

This attribute specifies the storage location for the subsequent downloaded ArcotID PKIs. A user interface is also provided for choosing the storage location. If the user specifies any option other than MEMORY, then the WalletInMemory attribute will be deprecated.

The following are the possible values or the combination of any:

- HD

  Stores the ArcotID PKI in the hard disk.
- USB

  Stores the ArcotID PKI in the USB flash drive.
- MEMORY

  Stores the ArcotID PKI in the memory for the current browser session.

CredentialFilter

This specifies how the clients should filter the credentials while querying an ArcotID PKI, during authentication or during any other use of an ArcotID PKI. Filtering criteria can include parameters such as, storage medium or issuing CA.

A credential filter is a set of expressions containing *<attribute><operator><value>*. Each expression is separated by an Ampersand (&).

For example, CertSubject=~OU%3DTesting&storagetype==hd is a credential filter, which displays only ArcotID PKIs that are stored in hard disk and containing the substring OU=Testing. Equal (=) signs that appear in the values need to be encoded as %3D.

There are four supported operators:

- == exact string match
- != not exact string match
- =~ substring match
- !~ not substring match

The case-sensitivity of the match is controlled by the case of the <*attribute*>. If it is all lower-case then the match is case-insensitive. Therefore in the example, the CertSubject match is case-sensitive while the storage type value is not.

The supported <*attribute*> values are:

- userid ArcotID PKI attribute
- org ArcotID PKI attribute
- alias ArcotID PKI attribute
- input_method ArcotID PKI attribute
- scrstyle ArcotID PKI attribute
- scrorder ArcotID PKI attribute
- ShowPinKeys Y/N - derived: input_method != keyboard
- ScramblePinPad Y/N (Y when scrstyle=once or always, N when scrstyle=never)
- ScrambleAlways Y/N (Y when scrstyle=always)
- ScrambleRandom Y/N (Y when scrorder=random)
- banner ArcotID PKI attribute
- bannerURL ArcotID PKI attribute
- domain ArcotID PKI attribute
- url_main ArcotID PKI attribute
- url_help ArcotID PKI attribute
- name same as the card name

- CardName name of card
- WalletName name of wallet
- CertSubject Arcot certificate attribute
- CertIssuerSubject Arcot certificate attribute
- CertSerialNumber Arcot certificate attribute
- CertNotBefore Arcot certificate attribute
- CertNotAfter Arcot certificate attribute
- KeyTypeCode key type of Arcot private key, for example, Arcot/RSA.
- cardImage base64-encoded ArcotID PKI attribute
- label ArcotID PKI attribute
- KeyFortEnabled Y/N (Y when ArcotID PKI attribute KeyFortURL is present)
- StorageType wallet attribute - device of locally stored ArcotID PKI - value domain (hd, mem, usb, cdrom)
- storage_type
- serialnumber ArcotID PKI attribute

The value portion of the expression should be URL-encoded if it contains any of the following special characters:

'~', '!', '=', '&','<', '>'

AID_LOOKUP_MODE

This attribute specifies how the clients should search the ArcotID PKI of the users in the database. The ArcotID PKIs are searched using username, orgname, alias, or a combination of these parameters.

Note: ArcotID PKI username aliases are stored as unsigned attributes inside the ArcotID PKI. These unsigned attributes are of the format Alias.<Application Context>=<alias>. For example, Alias.safebank-online=jdoe-sb.

The following lookup modes are supported:

- USERNAME_ONLY_MODE

  Clients searches the ArcotID PKI of the user by their username and orgname.
- ALIAS_ONLY_MODE

  Client searches the ArcotID PKI of the user by their alias and orgname.
- USERNAME_AND_ALIAS_MODE

In this mode, the client first searches the ArcotID PKI of the user by their username and orgname. If a matching ArcotID PKI is not found, then it searches the ArcotID PKI based on the orgname and alias.

Note: This is also the default mode.

DeviceLocking

This attribute configures the device locking mechanism in various ways. If no valid attributes are provided, then the default value all is used. Attributes are delimited by Underscore, when two or more are used.

The following are the possible values for this attribute:

- all

  Uses all the below mentioned device locking techniques.

- mem

  Physical memory size of the client’s machine.

- vol

  Identifies the partition that houses the volume to be locked. The volume is identified by the volume identifier.

- mac

  The distinctive address that identifies a Network Interface Card (NIC)is called the Media Access Control (MAC) address.

  A MAC address is a unique character string that identifies a specific physical device, which means one individual NIC. Therefore, the MAC address does not change for the life of the NIC. Because your NIC's MAC address is permanent, it is often referred to as the real or physical address of a computer.

- moth

  Mother board serial number and manufacturer name. This information is not always present.

- bios

  BIOS serial number and manufacturer name. This information is not always present.

- hd

  Hard Disk model number and manufacturer name. Only fixed hard disks are included but not the removable hard disks such as external USB or memory card. Changing or removing the hard disk will change the machine identifier.

- proc

  CPU information such as model number or clock speed. If the machine in use is a multiprocessor machine then the information from all the CPUs is included.

- enc

  Enclosure information is an unique information provided by the manufacturer, such as Service Tag provided by Dell for all its computers.

ScrambleStyle

This attribute facilitates the scrambling of pin pad, which is used to enter the ArcotID PKI password. The following are the different values used to set the frequency of scrambling:

- Never

  The pin pad is never scrambled.
- Once

  The pin pad is scrambled only once, when it is initially displayed. This is the default option.
- Always

  Pin pad is scrambled every time a key is clicked or pressed.

ScrambleOrder

This attribute defines the order in which the pin pad is scrambled. It is ignored if the password is entered using keyboard or the "ScrambleStyle" is set to Never. Following are the different values for this attribute:

- random

  Scrambling is done in random manner. For example, *8403172695*
- sequential

  Scrambling is done in sequential fashion. For example, *4567890123*

Returns

If the method is successful, then it returns a TRUE.

Example

```
var arcotClient = new ArcotClient();

// Set the PinPad scrambling order to SEQUENTIAL scrambling (shifting)
arcotClient.SetAttribute("ScrambleOrder", "sequential");

// Set the PinPad scrambling style to ALWAYS scramble
arcotClient.SetAttribute("ScrambleStyle", "always");
```

SetCurrentCardByIndex()

The client sets the current card to be the one with the given index in the current wallet.

Note: This is a deprecated function use ImportArcotID() (see page 67) instead.

Syntax

```
boolean SetCurrentCardByIndex(index)
```

Parameters

The following are the parameters of this method:

Parameter	**Type**	**Description**
index	integer	The index number of the card you wish to set. Usually "0" since there is only one card in a wallet.

Returns

If the method is successful, it returns TRUE. If the method is unsuccessful, it returns FALSE.

Example

```
var arcotClient = new ArcotClient();
var walletnameString = "GuestUser";
var b64walletString  = "MII0yAIBAAwHY3Rnb29kMTCCNDcwgjQzAgEBDAR
                        jYXJkoIHFMB8GCmCGSAGG+UYJAQEwEQQAAgEBAg
                        ILAgICAegCAgMYBIGhMIGeAgEAMA4GCmCGSA GG+
                    UYJAgAFAASBiDCBhQJBAMzNUdGcaBZsiAt/88Dk9...";

// Wallet will be saved to either hard disk or USB
arcotClient.SetAttribute("WalletInMemory", false);

// Set the current wallet
arcotClient.SetCurrentWalletFromEncoding(b64walletString);

// Set the card at index 0 as the current (and only) card in the wallet
var returnValue = arcotClient.SetCurrentCardByIndex(0);
if (returnValue)
{
    if (arcotClient.AddCurrentCardToWallet(walletnameString))
    {
        document.write("<P>Card added to wallet successfully</P>");
    }
    else
    {
        document.write("<P>Failed to add card to wallet</P>");
    }
}
```

SetCurrentWalletFromEncoding()

Sets the current wallet to be the wallet passed in as a string parameter. This function is typically called before calling AddCurrentCardToWallet() during a roaming download.

Note: This is a deprecated function use ImportArcotID() (see page 67) instead.

Syntax

```
boolean SetCurrentWalletFromEncoding(walletEncoding)
```

Parameters

The following are the parameters of this method:

Parameter	Type	Description
walletEncoding	string	A base-64 encoded string of the wallet you wish to set.

Returns

If the method is successful, then it returns TRUE. If the method is unsuccessful, then it returns FALSE.

Example

```
var arcotClient = new ArcotClient();
var walletnameString = "GuestUser";
var b64walletString  = "MII0yAIBAAwHY3Rnb29kMTCCNDcwgjQzAgEBDAR
                        jYXJkoIHFMB8GCmCGSAGG+UYJAQEwEQQAAgEBAg
                        ILAgICAegCAgMYBIGhMIGeAgEAMA4GCmCGSA GG+
                   UYJAgAFAASBiDCBhQJBAMzNUdGcaBZsiAt/88Dk9... ";

// Wallet will be saved to either hard disk or USB
arcotClient.SetAttribute("WalletInMemory", false);

// Set the current wallet
arcotClient.SetCurrentWalletFromEncoding(b64walletString);

// Set the card at index 0 as the current (and only) card in the wallet
var returnValue = arcotClient.SetCurrentCardByIndex(0);
if (returnValue)
{
    if (arcotClient.AddCurrentCardToWallet(walletnameString))
    {
        document.write("<P>Card added to wallet successfully</P>");
    }
    else
    {
        document.write("<P>Failed to add card to wallet</P>");
    }
}
```

AddCurrentCardToWallet()

Adds the current card to the named wallet.

Syntax

`boolean AddCurrentCardToWallet(`*`data`*`)`

Parameters

The following are the parameters of this method:

Parameter	Type	Description
data	string	Name of wallet.

Returns

If the method is successful, it returns TRUE. If the method is unsuccessful, it returns FALSE.

SignChallenge()

Important! This method is currently *not* supported for Signed Java Applet client.

Triggers the ArcotID PKI plug-in or applet to sign the challenge from the authentication server. User is provided with an interface to select the ArcotID PKI and enter the password.

Syntax

```
string SignChallenge(challenge)
```

Parameters

The following are the parameters of this method:

Parameter	**Type**	**Description**
challenge	string	Base-64 encoded challenge string that is to be signed.

Returns

Base-64 encoded string that includes the digital signature and the Arcot certificate, which contains the encrypted public key.

Discussion

When a client sends a signed challenge to AuthMinder, it is authenticated by the authentication server. If the server authenticates the challenge a response is generated.

The SignChallenge() method checks for this response. If a response is generated, then the method passes this response back to the client.

You obtain the challenge that is passed to this function by using one of the server-side APIs that are available for this purpose.

Example

```
var arcotClient = new ArcotClient();
var challengeString = "gCcBwHe/XkIxMjM0";

var response = arcotClient.SignChallenge(challengeString);
```

SignChallengeEx()

Triggers the plug-in or applet to sign a challenge string. The name of the ArcotID PKI and the password are passed in as parameters. The plug-in or applet does not present a GUI for this function.

Syntax

```
string SignChallengeEx(challenge, userID, cardname, PIN, orgName)
```

Parameters

The following are the parameters of this method:

Parameter	Type	Description
challenge	string	The encoded challenge string.
userID	string	The unique user identifier associated with the Arcot card.
cardname	string	The card name associated with the Arcot card.
PIN	string	The ArcotID PKI password.
orgName	string	The name of the AuthMinder organization to which the user belongs.

Returns

Base-64 encoded string that includes the digital signature and the Arcot certificate, which contains the encrypted public key.

Discussion

Use this method if you are creating a custom UI.

You obtain the challenge that is passed to this function by using one of the server-side APIs that are available for this purpose.

Example

```
var arcotClient     = new ArcotClient();
var challengeString = "gCcBwHe/XkIxMjM0";
var useridString    = "GuestUser";
var cardnameString  = "card";
var pinString       = "123456";
var orgName         = "Acme Vendor";
// Wallet will be saved to either hard disk or USB
var response = arcotClient.SignChallengeEx(challengeString,
                                           useridString,
                                           cardnameString,
                                           pinString, orgName);
```

ImportArcotID()

This method is used to download the ArcotID PKI from the server. The wallet name will be generated based on the internal wallet name of the ArcotID PKI. If the internal wallet name is not present, then the wallet's User ID will be used as the wallet name.

This single API call replaces the previous sequence of:

- Javascript call to SetAttribute("WalletInMemory", true | false). This indicates if the download will be temporary (memory only) or permanent (ArcotID PKI saved to disk).
- Javascript call to SetCurrentWalletFromEncoding(walletEncoding). This function takes the wallet encoding as a base-64 encoded string and saves it to memory as the current wallet.
- Javascript function, SetCurrentCardByIndex(0). This sets the "current card" to be the first card in the current wallet.
- Javascript function, AddCurrentCardToWallet(walletName). This stores the previously specified "current card" to a new wallet stored on the hard disk or memory. The parameter walletName is used as part of the file name for a wallet that is stored to disk.

In ArcotID PKI Javascript Client 6.0.4.3:

- To indicate to the API that the ArcotID PKI must be stored in the central device store, set the value of the StorageMode property to Shared in the props object that is passed to the API.
- To indicate that the ArcotID PKI must be stored in the user store, set the value of the StorageMode property to User.

Syntax for ArcotID PKI Javascript Client 6.0.4.3

```
boolean ImportArcotID(walletEncoding, StorageType, userName,
propertyObject)
```

Syntax for the Other Clients

```
boolean ImportArcotID(walletEncoding, StorageType, userName)
```

Parameters

The following are the parameters of this method:

Parameter	Type	Description
walletEncoding	string	Base-64 encoded string of the wallet to import.
StorageType	string	Specifies the medium, where the ArcotID PKI is stored. See "StorageType" for more information.

Parameter	Type	Description
userName	string	User name and alias, which are the unique user identifier associated with the Arcot card. **Note:** The alias is obtained from the user and stored in the ArcotID PKI as an attribute with the specified application context.
	property object	Contains the StorageMode property. The value of this property is set to either Shared or User to indicate that the ArcotID PKI must be stored in the central device store or the user store, respectively. If this property is not set, the user store is used as the default. **Note:** This parameter is available only in ArcotID PKI Javascript Client 6.0.4.3. It will be ignored by the other clients.

Returns

If the method is successful, then it returns TRUE. If the method is unsuccessful, then it returns FALSE.

Example for ArcotID PKI Javascript Client 6.0.4.3

```
var arcotClient = new ArcotClient();
var b64wallet = "MII0yAIBAAwHY3Rnb29kMTCCNDcwgjQzAgEBDAR
                 jYXJkoIHFMB8GCmCGSAGG+UYJAQEwEQQAAgEBAg
                 ILAgICAegCAgMYBIGhMIGeAgEAMA4GCmCGSA GG+
                 UYJAgAFAASBiDCBhQJBAMzNUdGcaBZsiAt/88Dk9... ";
var user = "jdoe-sb";
var props = {StorageMode:"Shared"};
// Import wallet and allow it to be saved on hard disk or memory only
var returnValue = arcotClient.ImportArcotID(b64wallet, "hd_memory", user, props);
```

Example for the Other Clients

```
var arcotClient = new ArcotClient();
var b64wallet = "MII0yAIBAAwHY3Rnb29kMTCCNDcwgjQzAgEBDAR
                 jYXJkoIHFMB8GCmCGSAGG+UYJAQEwEQQAAgEBAg
                 ILAgICAegCAgMYBIGhMIGeAgEAMA4GCmCGSA GG+
                 UYJAgAFAASBiDCBhQJBAMzNUdGcaBZsiAt/88Dk9... ";
var user = "jdoe-sb";
// Import wallet and allow it to be saved on hard disk or memory only
var returnValue = arcotClient.ImportArcotID(b64wallet, "hd_memory", user);
```

IsArcotIDAvailable()

This API checks whether the ArcotID PKI for the specified user is present.

Syntax

```
IsArcotIDAvailable(userName, orgname);
```

Parameters

Parameter	Type	Description
userName	string	User name and the alias, which are the unique user identifier associated with the Arcot card. **Note:** The alias is obtained from the user and stored in the ArcotID PKI as an attribute with the specified application context.
orgName	string	The name of the AuthMinder organization to which the user belongs.

Returns

If the method is successful, then it returns TRUE. If the method is unsuccessful, then it returns FALSE.

Example

```
var arcotClient = new ArcotClient();
var orgName = "safebank";

var userName = "jdoe-sb";

var response = arcotClient.IsArcotIDAvailable(userName, orgName);
```

IsArcotIDAvailableEx()

This API checks whether the ArcotID PKI for the specified user is present. The ArcotID PKI is searched based on the AID_LOOKUP_MODE attribute. The following table explains how the input parameters (userNameOrAlias, appctx, orgname) are interpreted and used by the 3 lookup modes:

Lookup Mode	Input Parameter Interpretation
USERNAME_ONLY_MODE	The userNameOrAlias is treated as username and the appctx is ignored. The ArcotID PKI is searched based on username and orgname.

Lookup Mode	Input Parameter Interpretation
ALIAS_ONLY_MODE	All three parameters are used to search an ArcotID PKI and the userNameOrAlias is treated as alias. If no matching ArcotID PKI is found, then an error code is returned. If orgname is NULL, then this parameter is ignored and the other two parameters are used to find a matching ArcotID PKI.
USERNAME_AND_ALIAS_MODE	The search is first based on USERNAME_ONLY_MODE, if the ArcotID PKI is not found then ALIAS_ONLY_MODE is used.

Syntax

```
isArcotIDAvailableEx(userNameOrAlias, appctx, orgname);
```

Parameters

Parameter	Type	Description
userNameorAliasID	string	User name and the alias, which are the unique user identifier associated with the Arcot card. **Note:** The alias is obtained from the user and stored in the ArcotID PKI as an attribute with the specified application context.
appctx	string	The name of the application context to which the user is logging in to. For example, the user with a single ArcotID PKI can access different portals (savings account, insurance, and credit card) of a bank account. Application context is supplied by the ArcotID Issuance application during ArcotID PKI issuance.
orgName	string	The name of the AuthMinder organization to which the user belongs.

Returns

If the method is successful, then it returns TRUE. If the method is unsuccessful, then it returns FALSE.

Example

```
var arcotClient = new ArcotClient();
var orgName = "safebank";
var appctx = "safebank-online";

var userAlias" = "jdoe-sb";

var response = arcotClient.isArcotIDAvailableEx(userAlias, appctx, orgName);
```

RemoveArcotID()

This method removes the ArcotID PKI with the specified wallet name and storage types. The function does a domain check of the calling Web page to ensure it matches the domain attribute stored in the ArcotID PKI. For the ArcotID PKI browser plug-in, this function will also un-register any key bag certificates that were registered in CAPI.

Syntax

```
boolean RemoveArcotID(walletname, StorageType, orgName)
```

Parameters

The following are the parameters of this method:

Parameter	Type	Description
walletname	string	The name of the wallet to be removed.
StorageType	string	Specifies the medium, where the ArcotID PKI is stored. See "StorageType" for more information.
orgName	string	The name of the AuthMinder organization to which the user belongs.

Returns

None.

Example

```
var arcotClient = new ArcotClient();
var walletname  = "GuestUser";
var orgName     = "Acme Vendor";

// Remove the ArcotID
var returnValue = arcotClient.RemoveArcotID(walletname, "hd", orgName);

// Reload all the ArcotIDs
arcotClient.RefreshArcotIDs();

// Get the number of wallets
var walletcount = arcotClient.GetGlobalAttribute("walletn.count");
```

RegisterCSPCertificates()

Important! This method is currently *not* supported for Signed Java Applet client.

For each ArcotID PKI, any key vault certificates will be registered into Microsoft CAPI (Crypto API) so that they can be used by Internet Explorer, Microsoft Outlook, and any other application that supports CAPI.

If ImportArcotID() was used to download the ArcotID PKI by using the Native Client, it is not necessary to invoke *RegisterCSPCertificates()*, because ImportArcotID() registers any key vault certificates if it has the permission to do so.

On the other hand, if an ArcotID PKI is imported to a machine using a USB token, RegisterCSPCertificates() needs to be invoked to ensure the key vault certificates are made available to CAPI.

Syntax

`string RegisterCSPCertificates(`*`walletname`*`)`

Parameters

The following are the parameters of this method:

Parameter	**Type**	**Description**
walletname	string	Wallet name of an ArcotID PKI whose certificates have to be registered.

Returns

None.

Example

```
var arcotClient = new ArcotClient();
var walletname = "GuestUser";

// Register the certificates in this ArcotID into MS CAPI
arcotClient.RegisterCSPCertificates(walletname);
```

SignChallengeNonBlocking()

Important! This method is currently *not* supported for Signed Java Applet client.

This function resolves GUI issues related to the Arcot applet, and enables to support the ATM GUI applet on Mac OS X. The API call receives callback functions for success and error handling.

Note: This function is not implemented in the native plugin. If called on the native plugin, the native plugin will return the error code ERR_FUNCTION_NOT_IMPLEMENTED.

Syntax

SignChallengeNonBlocking(*challenge, browser, onComplete, onFailure*)

Parameters

The following are parameters of the method:

Parameter	**Description**
challenge	The challenge that has to be signed.
browser	This option decides on where the callback functions should be invoked. This is an optional parameter. If set, the functions are invoked on a different window. If the value is null, the current browser is used.
onComplete	This is a JavaScript function that is invoked after the challenge has been successfully signed. The parameters passed are challenge and the signed challenge.
onFailure	A Javascript function that is invoked when the signing has failed. The parameters passed are challenge and the error message.

Example

```
arcotClient.SignChallengeNonBlocking(challenge, document, challengeSigned,

// Here we inline the error method, we could have called an external function instead

function (challenge, error) {
alert("Failed to sign the challenge, error: " + error);});

function challengeSigned(challenge, signedChallenge) {
// Set the signed challenge in the form
document.getElementByID("SignedChallenge").value = signedChallenge;
// Submit the form
document.getElementByID("LoginForm").submit();
}
```

GetVersionEx()

Gets the extended version information of the currently loaded Arcot plug-in or applet. In addition to the client version number, this function will also return the build number that is based on the current date. For example: "6.0 (200905211523)".

Syntax

```
string GetVersionEx()
```

Returns

The version of the ArcotID PKI Client installed on the client system.

Example

```
<HTML>
<HEAD>

<SCRIPT LANGUAGE="JavaScript">

var arcotClient = new ArcotClient();
var versionEx = arcotClient.GetVersionEx();
```

RefreshArcotIDs()

This API call will instruct the ArcotID PKI Client to re-read the storage areas for ArcotID PKIs. This is very useful for ArcotID PKIs stored on removable USB flash drives.

Syntax

```
RefreshArcotIDs()
```

Returns

None

Example

```
var arcotClient = new ArcotClient();

// Reload all the ArcotIDs
arcotClient.RefreshArcotIDs();

// Get the number of wallets
var walletcount = arcotClient.GetGlobalAttribute("walletn.count");
```

SignChallengeEx2()

This API selects an ArcotID PKI from a list of ArcotID PKIs available to it, based on the input parameters (userNameOrAlias, appctxName, orgName) and uses it to sign the incoming challenge. ArcotID PKI selection is based on the lookup mode that is set, see the "IsArcotIDAvailableEx() (see page 69)" section.

Note: The content and format of the signed challenge returned by SignChallengeEx2() is same as that returned by SignChallengeEx(). If you want to use the appctx and alias for logging, then you must pass this information to AuthMinder.

Syntax

```
SignChallengeEx2(challenge, PIN, userNameOrAlias, appctxName,
orgName)
```

Parameters

Parameter	**Type**	**Description**
challenge	string	The encoded challenge string.
PIN	string	The ArcotID PKI password.

Parameter	Type	Description
userNameorAlias	string	User name and the alias, which are the unique user identifier associated with the Arcot card. **Note:** The alias is obtained from the user and stored in the ArcotID PKI as an attribute with the specified application context.
appctxName	string	The name of the application context to which the user is logging in to. Application context is supplied by the ArcotID PKI Issuance application during ArcotID PKI issuance.
orgName	string	The name of the AuthMinder organization to which the user belongs.

Returns

Base-64 encoded string that includes the digital signature and the Arcot certificate, which contains the encrypted public key.

Example

```
var arcotClient = new ArcotClient();
var orgName = "safebank";
var appctx = "safebank-online";
var challengeString = "gCcBwHe/XkIxMjM0";
var userAlias" = "jdoe-sb";
var pinString = "123456";

var response = arcotClient.SignChallengeEx2(challengeString, pinString, userAlias,
appctx, orgName);
```

UpdateOfflineKeyBag()

This method is used to update the offline key bag of the user.

Syntax

```
boolean UpdateOfflineKeyBag(walletName, orgName, onlinePIN,
offlinePIN)
```

Parameters

The following are the parameters of this method:

Parameter	Type	Description
walletName	string	Wallet name of an ArcotID PKI for which the offline key bag has to be updated.

Parameter	Type	Description
orgName	string	The organization name to which the user belongs to.
onlinePIN	string	The ArcotID PKI password that is used to authenticate to the Key Authority server and fetch the latest key bag.
offlinePIN	string	The password that the user needs to use to access their certificates offline.

Returns

If the method is successful, then it returns TRUE. If the method is unsuccessful, then it returns FALSE.

Example

```
var arcotClient = new ArcotClient();
var  userIDStr  = "johnsmith";
var orgName = "Acme Vendor";
var onlinePINStr = "12345^";
var offlinePINStr = "123456"
var response = arcotClient.UpateOfflineKeyBag(userIDStr, orgName, onlinePINStr,
offlinePINStr);
```

RemoveArcotIDEx()

This API selects an ArcotID PKI from a list of ArcotID PKIs available to it, based on the input parameters (userNameOrAlias, appctxName, orgName) and deletes it. ArcotID PKI selection is based on the lookup mode that is set, see the IsArcotIDAvailableEx() (see page 69) section. After the ArcotID PKI is deleted it is not available for authentication.

Syntax

```
RemoveArcotIDEx(userNameOrAlias, appctx, orgname, storageType)
```

Parameters

Parameter	Type	Description
userNameorAlias	string	User name and the alias, which are the unique user identifier associated with the Arcot card. **Note:** The alias is obtained from the user and stored in the ArcotID PKI as an attribute with the specified application context.

Parameter	Type	Description
appctx	string	The name of the application context to which the user is logging in. Application context is supplied by the ArcotID Issuance application during ArcotID PKI issuance.
orgName	string	The name of the AuthMinder organization to which the user belongs.
storageType	string	Specifies the medium, where the ArcotID PKI is stored. See "StorageType" for more information.

Returns

If the method is successful, then it returns TRUE. If the method is unsuccessful, then it returns FALSE.

Example

```
var arcotClient = new ArcotClient();
var orgName = "safebank";
var appctx = "safebank-online";

var userAlias" = "jdoe-sb";

var storageType="hd";

var response = arcotClient.RemoveArcotIDEx(userAlias, appctx, orgName, storageType);
```

Chapter 7: Deploying the ArcotID PKI Client

This chapter describes the installation process for the ArcotID PKI Clients. The following topics are included in this chapter.

Note: Check for the supported operating environment, before performing the client installation. See "Operating Environment" for more information.

Deploying Flash Client

The ArcotID PKI flash client is in the ArcotIDClient.swf file. This file must be stored on a Web server or application server.

Place the ArcotIDClient.swf file in a directory anywhere on the Web server or application server. The URL of this directory must be added as the clientBaseURL attribute in the JavaScript that instantiates the client on the Web page.

Deploying Java Client

The following are the files for deploying different ArcotID PKI Java applets:

- ArcotApplet.jar.pack.gz

  Signed and packed applet

- ArcotApplet.jar

  Signed, unpacked applet

- ArcotAppletRaw.jar.pack.gz

  Unsigned and packed applet

- ArcotAppletRaw.jar

  Unsigned, unpacked applet

Perform the following steps to deploy these files:

1. Copy the above files in a directory anywhere on the Web server or application server.
2. Deploy the Pack200Servlet.class function that is supplied with the ArcotID PKI Client distribution package on the server where JARs are deployed.
3. Update the web.xml file of the Web application to redirect the applet request URLS to the servlet.

   The servlet analyses these requests, and based on the requesting client capabilities serves out either the packed jar or the unpacked jar.

Deploying Native Plugin

This section discusses the process to install ArcotID PKI browser plug-in on Windows and Mac operating systems.

- Deploying on Windows (see page 80)
- Deploying on Mac OS X (see page 83)

Deploying on Windows

The following two types of installation are covered for Windows:

- Web-Based Installation of Native Plug-In on Internet Explorer (see page 81)
- Local Installation of Native Plug-In on Internet Explorer (see page 81)
- Web-Based Installation of Native Plug-In on Mozilla Firefox (see page 82)

Web-Based Installation of Native Plug-In on Internet Explorer

The arcotplugin_win32.cab file contains the ArcotID PKI native client. This file must be stored on a Web server or application server.

Place the arcotplugin_win32.cab file in a directory anywhere on the Web server or application server. The URL of this directory must be added as the clientBaseURL attribute in the JavaScript that instantiates the client on the Web page.

Local Installation of Native Plug-In on Internet Explorer

The ArcotID PKI native client is also packaged as an EXE file that can be installed directly on a user's Windows operating system.

To install the ArcotID PKI Client on Windows:

1. Navigate to the appropriate directory and double-click the arcotplugin_win32.exe file.

   The Welcome to the ArcotID Client 6.2 Setup screen appears.

2. Click Next to proceed with the installation.

   The License Agreement screen appears.

3. Read the agreement carefully and select the I accept the terms in the license agreement option and click Install.

   The Installing ArcotID Client 6.2 screen appears and installs the software on the client’s machine with required settings and files, see "Installation Directory" (see page 85) for more information on the files.

   After the successful installation, Installation Complete screen appears.

4. Click Finish on this screen to exit the Wizard.

   The ArcotID Client Installer Information dialog appears.

5. Select Yes to restart the system.

Web-Based Installation of Native Plug-In on Mozilla Firefox

The arcotplugin.xpi file contains the ArcotID PKI native client. This file must be stored on a Web server or application server. Place the arcotplugin.xpi file in a directory anywhere on the Web server or application server. The URL of this directory must be added as the clientBaseURL attribute in the JavaScript that instantiates the client on the Web page.

When the user browses to the application pages, the arcotiplugin.xpi is automatically downloaded by the browser and opened for installation on the user machine.

Perform the following steps to install the ArcotID PKI Native Client plug-in on your system:

1. Click to install the new plug-in detected by the application.

   The Plugin Finder Service screen appears.

2. This dialog lists the ArcotID Client plug-in as x-arcotid-client. Click Manual Install to install the plug-in.

   The Software Installation screen appears.

3. Click Install Now in the Software Installation dialog.

   After successful installation, the Add-ons screen appears and lists the Arcot WebFort Client 6.2.

4. Click the Restart Firefox button to restart the Web browser.

Deploying on Mac OS X

Web-Based Installation of Native Plugin

The arcotplugin.dmg file contains the ArcotID native client. This file must be stored on a Web server or application server.

Place the arcotplugin.dmg file in a directory anywhere on the Web server or application server. The URL of this directory must be added as the clientBaseURL attribute in the JavaScript that instantiates the client on the Web page.

Installing the Native Plugin

When the user browses to the application pages, the arcotiplugin.dmg is automatically downloaded by the browser and opened for installation on the user machine.

Perform the following steps to install the ArcotID Native Client plug-in on your system:

1. Open the arcotplugin.dmg file to start the installation.

   The License Agreement screen.

2. Click the Agree button to accept the terms and proceed further.
3. The ArcotIDPluginClient drag-and-drop screen.
4. Drag the ArcotIDClientPlugin.plugin file and drop it in the Internet Plug-Ins directory by following the on-screen instructions.

   The ArcotID PKI Client plug-in is installed successfully.

Deploying JavaScript Client

The ArcotID PKI JavaScript client is in the arcotjsclient-jso.js file. This file must be stored on a Web server or application server.

Place the arcotjsclient-jso.js file in a directory anywhere on the Web server or application server. The URL of this directory must be added as the clientBaseURL attribute in the JavaScript that instantiates the client on the Web page.

Upgrading ArcotID PKI Clients

Flash Client

To upgrade the Flash client, you need to replace the ArcotIDClient.swf file available in the Web server or the application server with the latest ArcotIDClient.swf file.

Java Client

Upgrading is not necessary on a user's computer for the Java applet since the latest Java applet is downloaded to a user's computer each time they are invoked.

Native Client

To upgrade the ArcotID PKI native client, install the latest version of the native client. The installer for the new version will automatically remove the old client and installs the new client.

JavaScript Client

To upgrade the JavaScript client, you need to replace the arcotclient.js file available in the Web server or the application server with the latest arcotjsclient-jso.js file.

Uninstalling the Native Plug-In

This section lists the steps to uninstall ArcotID PKI Client:

- Uninstalling on Windows (see page 85)
- Uninstalling on Mac OS X (see page 85)

Note: The uninstallation process does not delete the ArcotID PKIs from the local machine. The user has to manually delete the .aid files.

Uninstalling on Windows

To uninstall the ArcotID PKI Client:

1. Navigate to the appropriate directory and double-click the arcotplugin_win32.exe file.

   The ArcotID Client Application Maintenance screen appears.

2. Click Next on the this screen to proceed with the uninstallation.

   The Program Maintenance screen appears.

3. Select the Remove option and click Next.

   The Remove the Program screen appears.

4. Click Back to change any of the settings or Remove to uninstall the software.

   After the software is uninstalled, the Installation Complete screen appears with a success message.

5. Click Finish to exit the wizard.

Uninstalling on Mac OS X

To uninstall the ArcotID PKI Client:

1. Navigate to the /Library/Internet Plugins folder.
2. Drag the ArcotIDPluginClient folder and drop it in the Trash folder.

Installation Directory

The ArcotID PKI Client software creates and modifies files on a user's computer. This section lists the files for Windows and Mac operating systems.

Windows Installation Directory

The following table lists the files in the ArcotID PKI Client directory for Windows:

Destination	**Client Type**	**Files**
<*%SYSTEMDRIVE%*>:\Program Files\Arcot Systems\ArcotID Client	Native Client	■ LICENSE.txt License information for the ArcotID PKI Client. ■ ArcotCSPInstall.exe Program that is used to register the Arcot CSP. ■ ArcotClient.sig Signature file that is used when the Arcot CSP is registered. ■ ArcotClient.dll DLL containing both the Arcot plug-in and Arcot CSP.
<*%SYSTEMDRIVE%*>:\Program Files\Common Files\Arcot Shared	Native Client	Folders, ■ conf Default configuration information for the Arcot client. ■ i18n Contains folders for the supported international languages. Each of these folders contains anc.i18n file, which contains the localized data for the respective language. ■ images Image files used by ArcotID PKI Client.
Windows\System32	Native Client	■ ArcotATMGUI.dll DLL used by the ArcotID PKI Client. ■ ArcotCardMgrGUI.dll DLL used by the ArcotID PKI Client. ■ ArcotOfflineTool.exe Executable file of Arcot Offline Tool. ■ ArcotPK11.dll Arcot PKCS#11 module. ■ ArcotProgressGUI.dll DLL used by the ArcotID PKI Client.

Destination	Client Type	Files
%APPDATA%\arcot\	Native Client and Signed Java Applet	In each user's "%APPDATA%" directory arcot directory is created. It contains: ■ conf/WebClient.ini Contains the user preferences. ■ ids Directory where ArcotID PKI files (.aid files) are stored.

Mac OS X Installation Directory

The following table lists the files in the ArcotID PKI Client directory for Mac OS X:

Destination	Client Type	Files
/library/Internet Plug-Ins/ArcotIDPluginClient/contents	Native Client	■ MACOS/ARcotIDClientPlugin File containing the Arcot plug-in. ■ Resources ■ info This file is read by the Web browser to collect the display information.

Registry Changes

The Arcot Native client software modifies the register on user's computer. The following new registry keys are created.

- HKEY_CURRENT_USER\Software\Arcot Systems, which is used to store the user-specific settings.
- HKEY_LOCAL_MACHINE\SOFTWARE\Arcot Systems, which is used to store the settings that apply to all users.

Chapter 8: ArcotID PKI As a Software Smartcard

In addition to being used for strong addition, ArcotID PKI can also be used to securely store the Open PKI keys and certificates. These keys are typically used for different applications or operations such as, email signing (S/MIME), document signing, and certificate-based authentication (open PKI).

The location where the open PKI keys and certificates are stored in the ArcotID PKI is called *key bag* or *key vault*. The key to access the secure key bag is referred to as Key Authority Key (KA Key) and is stored in the AuthMinder database.

To use the private keys that are stored in the key bag, the ArcotID PKI Client makes a request for the KA Key to the AuthMinder Server by signing the request with the camouflaged ArcotID PKI password. The AuthMinder Server authenticates the request, and then sends the KA Key to the client, which uses this key to open the key bag and access the private keys. This mode of accessing the private keys is known as accessing keys *Online*.

From this release, ArcotID PKI Client enables users to access their private keys *offline*, which means, to access the private keys, the ArcotID PKI Client need not connect to the AuthMinder Server to authenticate users. This feature helps users to use their private keys even if they do *not* have access to the network. For example, a roaming user with no connectivity to their corporate network.

To support this feature, the ArcotID PKI Native Client is shipped with the Arcot Offline Tool, which is a utility that is used for accessing the private keys offline. This tool is installed as part of ArcotID PKI Native Client installation, and is available in the system tray of the user's computer.

This chapter contains the following topics:

How It Works

The process of accessing the private keys offline is almost similar to accessing the keys when you are connected to the network. To enable the offline access, the user will have to set a new password (known as *offline password*) by using the Arcot Offline Tool. After registering the offline password, the Arcot Offline Tool creates a copy of the user’s key bag (offline key bag) and stores it in the ArcotID PKI.

Offline password enables secure access for you users to the network even if the application cannot access the network server.

After this initial setup, the users can use their private keys stored in the offline key bag.

Note: This workflow assumes that the ArcotID PKI credential has already been issued for the user.

The following workflow provides an overview of steps involved in accessing the keys offline:

1. The user connects to your network and installs the ArcotID PKI Native Client on their system.

   Note: The Arcot Offline Tool is *also* installed on the user’s system.

2. The user downloads the ArcotID PKI to their system by following your corporate policies.
3. The user starts the Arcot Offline Tool and provides the following information:
   - Enters the ArcotID PKI password.
   - Sets the offline password for accessing keys offline.
   - Re-confirms the offline password.
4. The Arcot Offline Tool authenticates the user.

   If the authentication was successful, then the Arcot Offline Tool unencrypts the key bag with the shared secret, encrypts it with the offline password, and then updates the ArcotID PKI with the key bag that is encrypted with offline password.

The user’s ArcotID PKI is embedded with the offline key bag that contains their private keys.

Key Features

This section lists the key features of offline access mode:

- Configurable Access Mode (see page 91)
- Device Locking (see page 92)
- N-Strikes (see page 93)
- Configurable Password Format (see page 93)
- Configurable Validity Period (see page 94)
- Expiry Notification (see page 95)
- Password Caching (see page 95)
- Customizable User Interface (see page 96)

The key features that are discussed in this section are set by using the ArcotID PKI attributes. To enable these features, you have to set the ArcotID PKI attributes as name-value pairs in the ArcotID PKI profile configuration supported by the AuthMinder Server.

Note: Refer to the "Configuring ArcotID PKI Credential Profile" section in the *CA AuthMinder Administration Guide* for more information on how to set the ArcotID PKI attributes in the ArcotID PKI profile.

Configurable Access Mode

The private keys can be accessed online or offline, the mode that is used to access these keys is determined by the AllowOfflineUseOfKeyBag ArcotID PKI attribute. This attribute determines whether the user is permitted offline access to their key bag. If the value of this attribute is set to True, then user can register their offline password, create an offline key bag, and then can access their keys and certificates offline.

The following table lists the values that are supported by the AllowOfflineUseOfKeyBag attribute:

Value	Description
TRUE	Specify this value if you want to enable the user to have offline access to their signing and encryption keys. If AllowOfflineUseOfKeyBag is set to TRUE, then the users must set their offline password by using the Arcot Offline Tool, and use this password to access the offline key bag. **Note:** By default, offline access is enabled.

Value	Description
FALSE	Specify this value if you do not want to enable offline access to users, which means users must be online (connected to the server) to access their signing and encryption keys. If AllowOfflineUseOfKeyBag is set to FALSE, then the user has to perform the regular ArcotID PKI authentication for accessing keys.

Device Locking

The Device Locking feature enables an offline key bag to be *locked* to a specific machine, so that the offline ArcotID PKI is not usable if it is copied to another machine.

The feature works by camouflaging (protecting) an offline key bag using a password made of two components.

1. The offline password selected by the user for the offline ArcotID PKI.
2. A new Machine PIN, which is derived from unique machine-specific information derived from the hardware characteristics of the client machine. Refer to the "Device Locking" (see page 26) section for more information on the machine parameters that are used for deriving machine PIN.

The device locking is done at the time of offline password registration during offline key bag creation. After an offline ArcotID PKI is locked to the user's machine, it is not usable if you copy it to another machine.

The offline_devlock_required ArcotID PKI attribute specifies whether the offline ArcotID PKI has to be locked to the device.

The following table lists the values that are supported by the offline_devlock_required attribute:

Value	Description
Yes	Specify this value if you want to lock the user's offline ArcotID PKI to their system.
No	Specify this value if you want to permit the users to copy their offline ArcotID PKI to another system and authenticate using the copied offline ArcotID PKI. **Note:** This value is selected by default.

If you enable device locking, then you have to use the offline_devlock_type attribute to specify the locking parameters. You have to pass the device locking parameters in a string format.

The supported parameters and the specification format are same as that of the devlock_type attribute used for locking the online ArcotID PKI. Refer "Device Locking" (see page 26) for more information.

N-Strikes

You can configure the number of times a user is permitted to enter the offline password incorrectly to access their offline ArcotID PKI. If the user provides the wrong password for the configured number of times, then their account is locked and the offline key bag is disabled. To re-enable the key bag, the user has to synchronize the offline key bag by using the Arcot Offline Tool.

The SuccessiveFailedOfflineLoginAttempts ArcotID PKI attribute specifies the number of times the user can provide incorrect offline password. Set the value of this attribute to the number of times the user is allowed to enter incorrect password.

Note: This feature is disabled by default.

Configurable Password Format

You can also enforce whether the offline password must be different from the online ArcotID PKI password. The EnforceDifferentOfflinePasword ArcotID PKI attribute specifies this feature.

The following table lists the values supported by the EnforceDifferentOfflinePassword attribute:

Value	Description
True	Specify this value if you want the offline password to be different from the online ArcotID PKI password.
False	Specify this value if you do not want to restrict the offline password to be different from the online ArcotID PKI password. **Note:** This value is selected by default.

To configure the offline password format, you need to use the OfflinePasswordProfile ArcotID PKI attribute. The effectiveness of password is determined by a combination of the length of the password, and number of numerals and special characters in it.

The following table lists the options that the OfflinePasswordProfile ArcotID PKI attribute provides to specify the offline ArcotID PKI password characteristics:

Format	**Default Value**	**Description**
minlen	4	Specifies the minimum length of the password.
maxlen	8	Specifies the maximum length of the password.
minsplchars	0	Specifies the minimum number of special characters required in the password. **Note:** All special characters excluding ASCII characters (0-31) are supported.
minnumchars	0	Specifies the minimum number of numerals required in the password.

You have to specify the password characteristics in a string format as shown below:
`"minlen=<n>;maxlen=<n>;minsplchars=<n>;minnumericchars=<n>"`

Note: If you specify the format names incorrectly or provide non-numeric values, then the default values are used.

Configurable Validity Period

You can configure the validity period of an offline key bag by using the OfflineKeyBagExpiryDays attribute. Set the value of this attribute to the number of days for which the offline key bag must be valid.

The offline key bag is blocked at the completion of the validity period. To unblock the key bag, the user has to synchronize the offline key bag by using the Arcot Offline Tool.

Expiry Notification

If the user’s offline ArcotID PKI key bag is about to expire, then you can notify the user regarding their impending offline ArcotID PKI expiry, and enforce them to register for the offline access again. The new ArcotID PKI is downloaded to the location that is configured using the ArcotIDLocation parameter in the WebClient.ini file.

The OfflineKeyBagExpiryWarningDays ArcotID PKI attribute specifies the number of days before the user receives the offline key bag expiry warning. You have to set the value of this attribute to the number of days before key bag expiry from when the user starts to receive the expiry warning. The warning message is delivered to the user through the Arcot Offline Tool.

For the Arcot Offline Tool to display the ArcotID PKI expiry warning balloon, you need to configure the OfflineKeyBagExpiryWarningInterval parameter in the [arcot/WebClient/AOT] section of WebClient.ini file. The OfflineKeyBagExpiryWarningInterval parameter defines the interval at which the expiry warning balloon appears on the user’s system. If OfflineKeyBagExpiryWarningInterval is set to 2, then the balloon appears once every two hours. By default, OfflineKeyBagExpiryWarningInterval is set to 1.

To reactivate the expired key bag, the user has to synchronize the offline key bag by using the Arcot Offline Tool.

Password Caching

By default, the online and offline passwords are cached so that the user need not enter the password for every security operation. This cached password can be used any number of times.

You can configure the period and the maximum number of times the cached password can be used. After the validity period expires or the cached password is used for the maximum number of times configured, the password stored in the cache is deleted. For any subsequent operations related to the key bag access, the users will be prompted for their password.

The following table lists the ArcotID PKI cache attributes for online and offline passwords:

Attribute	**Description**
OnlinePasswordCacheExpiryMinutes	This ArcotID PKI attribute specifies the minutes for which the cached password can

Attribute	Description
OfflinePasswordCacheExpiryMinutes	be used. You have to set the value of this attribute to the period that you want to use the cached value. Supported Values: ■ **<n>**: Indicates the minutes for which the cached password can be used.
OnlinePasswordCacheNumberOfUse	This ArcotID PKI attribute specifies the number of times the cached password can be used. You have to set the value of this attribute to the number of times you want your users to perform key bag-related operations without entering their password. Supported Values: ■ **<n>**: Indicates the number of times the cached password can be used. ■ **0**: Indicates that the cached password can be used any number of times. ■ **-1**: Indicates that the password *cannot* be cached, and the user must enter their password for every key bag-related operation.
OfflinePasswordCacheNumberOfUse	

Customizable User Interface

The graphical user interface (GUI) of the Arcot Offline Tool can be completely customized. All GUI elements such as, button headings, labels, success and error messages can be customized according to your requirement.

Customizing the GUI

Perform the following steps to customize the GUI elements:

1. Navigate to the <%SYSTEMDRIVE%>:\Program Files\Common Files\Arcot Shared\conf folder.
2. Open the WebClient.ini file in a text editor.
3. Configure the parameters in the [arcot/WebClient/AOT] section, as explained in the following table:

Parameter	Description
EnableDescriptionInDialog	Indicates that a description can be included in the Arcot Offline Tool dialogs. By default, the value of this parameter is set to False.

Parameter	**Description**
BackgroundColor	Indicates the background color (in RGB format) that is used in the Arcot Offline Tool dialogs. By default, the value of this parameter is set to R=212;G=208;B=200.
CombineChangePassword AndFYP	Indicates that instead of having **Change Password** and **Forgot Your Password** dialogs separately, you can have only one dialog. By default, the value of this parameter is set to False.
ArcotOfflineToolIcon	Indicates the file corresponding to the Arcot Offline Tool icon that appears in the Windows System Tray. This file must be in .ico format, and the file dimensions must be 16 pixels along the X- and Y-axis. You need to include this file in the following folder: <%SYSTEMDRIVE%>:\Program Files\Common Files\Arcot Shared\images
ArcotOfflineToolLogo	Indicates the logo that appears in the Arcot Offline Tool dialogs. The dimensions of this file must be 55 pixels along the X-axis and 59 pixels along the Y-axis. You need to include this file in the following folder: <%SYSTEMDRIVE%>:\Program Files\Common Files\Arcot Shared\images

4. Save and close the file.

Customizing the Text

The Arcot Offline Tool sources all the GUI text from the anc.i18n file. Perform the following steps to change the GUI labels stored in this file:

1. Navigate to the <%SYSTEMDRIVE%>:\Program Files\Common Files\Arcot Shared\i18n\<*Locale_name*> folder.
2. Open the anc.i18n file in a text editor.
3. Make the necessary changes.
4. Save and close the anc.i18n file.

Support for Open Standards

The ArcotID PKI Client supports standard Microsoft Cryptographic Service Provider (CSP) APIs and PKCS#11 to securely use the Open PKI keys stored in the ArcotID PKI. This section lists the CSP and PKCS#11 files related to this operation:

- Cryptographic Service Provider (see page 98)
- PKCS#11 Module (see page 99)

Cryptographic Service Provider

The ArcotClient.dll file available in the <%SYSTEMDRIVE%>:\Program Files\Arcot Systems\ArcotID Client folder provides the Arcot CSP functionality. This library file is signed by Microsoft as a CSP that is trusted by the Windows operating system.

If the user performing the installation has appropriate permission, then the Arcot CSP is registered by the Arcot Client installer automatically. If the Arcot CSP is not registered during installation, then use the ArcotCSPInstall.exe tool available in the <%SYSTEMDRIVE%>:\Program Files\Arcot Systems\ArcotID Client to register the CSP at a later time.

PKCS#11 Module

The ArcotPK11.dll library file available in the <%SYSTEMDRIVE%>:\WINDOWS\system32 folder provides the Arcot PKCS#11 functionality. This file is recognized by most applications that support PKCS#11 modules. For example, Mozilla Firefox, Adobe Reader, and Adobe Acrobat.

The Arcot PKCS#11 module conforms to PKCS#11 version 2.01. The Arcot PKCS#11 module does not support all PKCS#11 API functions. Below is the list of PKCS#11 functions that are supported by the Arcot module:

- C_Initialize
- C_Finalize
- C_GetInfo
- C_GetFunctionList
- C_GetSlotList
- C_GetSlotInfo
- C_GetMechanismList
- C_GetTokenInfo
- C_GetMechanismInfo
- C_OpenSession
- C_CloseSession
- C_CloseAllSessions
- C_GetSessionInfo
- C_Login
- C_Logout
- C_CreateObject
- C_GetAttributeValue
- C_SetAttributeValue
- C_FindObjectsInit
- C_FindObjects
- C_FindObjectsFinal
- C_DecryptInit
- C_Decrypt
- C_SignInit
- C_Sign
- C_Verify

- C_VerifyInit
- C_GenerateKey
- C_GenerateKeyPair

The following PKCS#11 functions are not supported by the Arcot module:

- C_InitToken
- C_InitPIN
- C_SetPIN
- C_GetOperationState
- C_SetOperationState
- C_CopyObject
- C_DestroyObject
- C_GetObjectSize
- C_EncryptInit
- C_Encrypt
- C_EncryptUpdate
- C_EncryptFinal
- C_DecryptUpdate
- C_DecryptFinal
- C_DigestInit
- C_Digest
- C_DigestUpdate
- C_DigestKey
- C_DigestFinal
- C_SignUpdate
- C_SignFinal
- C_SignRecoverInit
- C_SignRecover
- C_VerifyFinal
- C_VerifyUpdate
- C_VerifyRecoverInit
- C_VerifyRecover
- C_DigestEncryptUpdate

- C_DecryptDigestUpdate
- C_SignEncryptUpdate
- C_DecryptVerifyUpdate
- C_WrapKey
- C_UnwrapKey
- C_DeriveKey
- C_SeedRandom
- C_GenerateRandom
- C_GetFunctionStatus
- C_CancelFunction
- C_WaitForSlotEvent

Arcot has defined vendor-specific codes as permitted in the PKCS#11 standard. These codes are used in the Arcot PKCS#11 module.

Vendor-Specific PKCS#11 Mechanisms

CKM_ARCOT_SIGN_CHALL 0x87000000

This mechanism is:

- Returned by C_GetMechanismList
- Supported by C_GetMechanismInfo
- Used as argument to C_SignInit to request an Arcot proprietary-format signature by using the Arcot Private Key. The returned signature format is the DER encoding of SEQUENCE{ version, Certificate, X509Signature}.

Vendor-Specific PKCS#11 Attributes

The below attributes are supported by C_GetAttributeValue():

- CKA_ARCOT_CARD_IMAGE 0x87000000
- CKA_ARCOT_WALLET_NAME 0x87000001
- CKA_ARCOT_CARD_NAME 0x87000002
- CKA_ARCOT_WALLET_PATH 0x87000003
- CKA_ARCOT_CARD_DOMAIN 0x87000005

The below attribute is only supported when the object is an internal Arcot certificate:

- CKA_ARCOT_CARD_DOMAIN: card attribute domain

The below attributes are only supported when the object is a key vault certificate stored in an ArcotID PKI:

- CKA_ARCOT_CARD_IMAGE: card attribute banner
- CKA_ARCOT_WALLET_NAME: ArcotID PKI name
- CKA_ARCOT_CARD_NAME: ArcotID PKI card name
- CKA_ARCOT_WALLET_PATH: Configuration value for the folder where ArcotID PKIs are stored by default.

Vendor-Specific Flags

The following are the vendor specific flags used by Arcot PKCS#11 module:

- CKF_ARCOT_NO_KEY_AUTHORITY_SESSION 0x01000000

  Supported as an argument to C_OpenSession. Indicates that no attempt to login to the key authority server should be made. This means the slot can be used for functions with the Arcot Certificate only, and the certificates in the card's key vault (if any) will not be visible.

- CKF_ARCOT_ONLY_KEY_AUTHORITY_SESSION 0x02000000

  Supported as an argument to C_OpenSession. Indicates that operations in the session will only refer to certificates in the card's key bag. The Arcot Certificate will not be visible. This flag is ignored if CKF_ARCOT_NO_KEY_AUTHORITY_SESSION is present.

- CKF_ARCOT_TOKEN_KEYFORT_ENABLED 0x04000000

  Returned by C_GetTokenInfo for a given slot when the slot represents and ArcotID PKI that contains a Key Store.

Chapter 9: ArcotID PKI As a Software Smartcard: End User Experience

ArcotID PKI is a software smartcard that your users can use to securely store their Open PKI keys and certificates. These keys are typically used for different applications or operations such as, email signing (S/MIME), document signing, and certificate-based authentication (open PKI).

Depending on the security settings defined by you (credential provider), your users will be allowed to access their certificates either by connecting to the network and authenticating to a network server (also known as *online access*) or your users will be allowed to access them after providing an offline password that gets verified locally (this is also known as *offline access*). Offline Access is typically required by travelling employees, remote workers, and field personnel.

The Arcot Offline Tool utility can be used to register the offline password for offline access. This utility is installed on users system as part of ArcotID PKI Native Client installation.

This chapter covers the following topics:

- Installing ArcotID PKI Native Client (see page 104)
- Working With Arcot PKI Offline Tool (see page 104)
- Working With Microsoft Outlook Emails (see page 109)
- Working With Adobe PDF Documents (see page 113)

Installing ArcotID PKI Native Client

End users need to perform the following steps to install the ArcotID PKI Native Client:

1. Navigate to the appropriate directory and double-click the arcotplugin_win32.exe file.

   The Welcome to the ArcotID Client 6.2 Setup screen appears.

2. Click Next to proceed with the installation.

   The License Agreement screen appears.

3. Read the agreement carefully and select the I accept the terms in the license agreement option and click Install.

   The Installing ArcotID Client 6.2 screen appears and installs the software on the client's machine with required settings and files.

   After the successful installation, Installation Complete screen appears.

4. Click Finish on this screen to exit the Wizard.

   The ArcotID Client Installer Information dialog appears.

5. Select Yes to restart the system.

The ArcotID Native Client installer installs the Arcot Offline Tool on the user's system. This tool will be available in their system tray with the Arcot Offline Tool entry.

If the tool is not available in the system tray, then the user can navigate to the <%SYSTEMDRIVE%>:\WINDOWS\system32 folder and double-click ArcotOfflineTool.exe to include it in the system tray.

Working With Arcot Offline Tool

This section provides an overview of tasks that users can perform by using the Arcot Offline Tool. Following topics are covered in this section:

- Registering for Offline Access (see page 105)
- Synchronizing ArcotID PKI Content (see page 106)
- Changing Offline Password (see page 107)
- Forgot Your Password (see page 108)
- Deleting Offline Password (see page 108)
- Checking the Access Mode (see page 109)
- Checking the Arcot Offline Tool Version (see page 109)

Registering for Offline Access

To use the private keys offline, users need to first set up their offline password. This password provides them offline access to their certificate and keys.

End users need to perform the following steps to set up the offline password:

1. Right-click the Arcot Offline Tool available in the system tray.
2. Select the Register Offline Password option.

   The Register Offline Password for ArcotID dialog appears.
3. Enter the following information in this dialog:
   - Online Password: ArcotID PKI password.
   - New Offline Password: The password that will be used to access the ArcotID PKI offline.
   - Confirm Offline Password: Re-enter the offline password. This value must be same as that entered in the New Offline Password field.
4. Click OK to set the Offline password.

   If the offline password registration was successful, then the dialog with the "Offline password registration was successful" message appears.

   Note: If the user enters a wrong online password or different offline passwords in Step 3, then an appropriate error message appears. In this case, the user must repeat Step 1 to Step 4.
5. Click OK to continue.

If the registration was successful, then it indicates that the offline access setup was successful. Based on the security settings as configured by you, your users might be prompted for the offline password any time they perform security operations, such as email signing, email decryption, or PDF document signing.

Refer to the "Working With Microsoft Outlook Emails" (see page 109) and "Working With Adobe PDF Documents" (see page 113) sections for more information on how to use the private keys and certificates stored in the ArcotID PKI for opening encrypted emails and signing PDF documents.

Synchronizing ArcotID PKI Content

If user’s certificates are renewed or new certificates are issued to them, then you must notify them to synchronize their credentials for offline access.

Users need to perform the following steps to synchronize their credentials for offline access:

1. Right-click the Arcot Offline Tool available in the system tray.
2. Select the Synchronize Offline Credentials option.

   The Synchronize Offline Credentials dialog appears.
3. Enter the following information in this dialog:
   - Online Password: ArcotID PKI password.
   - Offline Password: The password that is used to access the ArcotID PKI offline.
4. Click OK to synchronize the ArcotID PKI.

   If the password synchronization was successful, then "Credentials synchronization was successful" message appears.

   Note: If the user enters a wrong online or offline password in Step 3, then an appropriate error message appears. In this case, the user must repeat Step 1 to Step 4.
5. Click OK to continue.

   After successful synchronization, the latest certificates and keys are available on the user’s system for their use.

Changing Offline Password

Users need to perform the following steps to change their offline password:

1. Right-click the **Arcot Offline Tool** available in the system tray.
2. Select the **Change Password** option.

   The Change Offline Password dialog appears.
3. Enter the following information in this dialog:
   - **Online Password**: The current password used for the online access.
   - **New Offline Password**: The new password to be set for offline access.
   - **Confirm Offline Password**: Re-enter the new offline password. This value must be same as that entered in the **New Offline Password** field.
4. Click **OK** to change the offline password.

   If the password change was successful, then "Change Offline password was successful" message appears.

   Note: If the user enters a wrong offline password in Step 3, then an appropriate error message appears. In this case, the user must repeat Step 1 to Step 4.
5. Clicks **OK** to continue.

Forgot Your Password

If the users forget their offline password, then they can set a new offline password by using the Arcot Offline Tool.

Users need to perform the following steps to reset their offline password:

1. Right-click the Arcot Offline Tool available in the system tray.
2. Select the Forgot Your Password? option.

   The Forgot Your Password dialog appears.
3. Enter the following information in this dialog:
   - Online Password: ArcotID PKI password.
   - New Offline Password: The password that is used for offline access.
   - Confirm Offline Password: Re-enter the new offline password. This value must be same as that entered in the Confirm Offline Password field.
4. Click OK to reset the offline password.

   If the password reset was successful, then "Offline password registration was successful" message appears.

   Note: If the user enters a wrong online password in Step 3, then an appropriate error message appears. In this case, the user must repeat Step 1 to Step 4.
5. Click OK to continue.

Deleting Offline Password

Arcot Offline Tool provides an option to delete the offline credentials that have been created on the user’s system. Users need to perform the following steps to delete their offline password:

1. Right-click the Arcot Offline Tool available in the system tray.
2. Select the Delete Offline Credentials option.

   The confirmation dialog appears.
3. Click OK to delete the ArcotID PKI offline password.

Checking the Access Mode

Arcot Offline Tool provides an option for your users to know the access mode that is in effect for them, and the password that they are required to use to access their certificates.

Users need to perform the following steps to check their access mode:

1. Right-click the **Arcot Offline Tool** available in the system tray.
2. Select the **Status** option.

   Based on your corporate security policies and the access mode that is configured for you, the dialog that appears indicates one of the following:

   ■ You have online access to your credentials. You are prompted for online password when you perform security operations.

   ■ You have offline access to your credentials. You are prompted for offline password when you perform security operations.
3. Click **OK** to continue.

Checking the Arcot Offline Tool Version

Users need to perform the following steps to check the version of the Arcot Offline Tool installed on their system:

1. Right-click the **Arcot Offline Tool** available in the system tray.
2. Select the **About Arcot Offline Tool** option.

   The About AOT dialog indicating the Arcot Offline Tool version appears.
3. Click **OK** to continue.

Working With Microsoft Outlook Emails

This section describes how to use the private keys and certificates stored in ArcotID PKI to sign, and encrypt and decrypt Microsoft Office Outlook emails. It covers the following topics:

■ Signing Emails (see page 110)

■ Encrypting Emails (see page 111)

■ Opening Encrypted Emails (see page 113)

Signing Emails

Users need to perform the following steps to sign a Microsoft Office Outlook 2010 email by using their certificates stored in ArcotID PKI:

Note: The following steps are applicable to Microsoft Office Outlook 2010 *only*. Refer to vendor documentation for information on how to perform these tasks using other versions.

1. Start Microsoft Office Outlook.
2. Activate the Home tab.
3. Click the New E-mail option.

   The email message appears.
4. In the Options tab, in the More Options group, click the Message Options Dialog Box Launcher.

   The Properties dialog appears.
5. Click Security Settings.

   The Security Properties dialog appears.
6. Select the Add digital signature to this message check box, and in the Security Settings section click Change Settings.

   The Change Security Settings dialog appears.
7. In the Certificates and Algorithms section, click the Choose button against the Signing Certificate field.

   The Select a Certificate dialog appears.
8. Select the certificate that must be used for signing, and click OK.

   The certificate is now configured for digital signing.
9. Click OK in the Change Security Settings dialog.
10. Click OK in the Security Properties dialog.
11. Click Close in the Properties dialog.
12. Draft the message and enter the email ID of the recipient.
13. Click Send to the send the email.

    Based on your corporate security policies and the access mode that has been configured for you, the *ArcotID PKI Client 6.2 Password* dialog indicates one of the following:

    - You have been granted online access to your credentials, and have to provide your online password.
    - You have been granted offline access to your credentials, and have to provide your offline password.

14. In the Password field, enter the corresponding ArcotID PKI password.
15. Click OK to submit the password.

    The ArcotID PKI Client verifies the user’s password. If the authentication was successful, then the email is digitally signed using user’s certificate and sent to the intended recipient.

Encrypting Emails

This section lists the following topics:

- Performing Initial Configuration (see page 111)
- Sending Encrypted Emails (see page 112)

Performing Initial Configurations

To successfully encrypt email messages, users must complete the following prerequisite tasks:

- Procure the digital certificate of the user to whom the encrypted email has to be sent.
- Create a contact for the intended email recipient in the My Contacts list, and include the certificate of the recipient in that contact.

Sending Encrypted Emails

Users must perform the following steps to encrypt a Microsoft Outlook 2010 email by using their certificates stored in ArcotID PKI:

Note: The following steps are applicable to Microsoft Outlook 2010 *only.* Refer to vendor documentation for information on how to perform these tasks using other versions.

1. Start Microsoft Office Outlook.
2. Activate the Home tab.
3. Click the New E-mail option.

   The email message appears.
4. In the Options tab, in the More Options group, click the Message Options Dialog Box Launcher.

   The Properties dialog appears.
5. Click Security Settings.

   The Security Properties dialog appears.
6. Select the Encrypt Message Contents and Attachments check box, and in the Security Settings section click Change Settings.

   The Change Security Settings dialog appears.
7. In the Certificates and Algorithms section, click the Choose button against the Encryption Certificate field.

   The Select a Certificate dialog appears.
8. Select the certificate for encrypting the email, and click OK.

   The recipient’s certificate is configured for encrypting email.
9. Click OK in the Change Security Settings dialog.
10. Click OK in the Security Properties dialog.
11. Click Close in the Properties dialog.
12. Draft the message and enter the email ID of the recipient.
13. Click Send to the send the encrypted email.

Opening Encrypted Emails

Users must perform the following steps to open an encrypted email.

1. Start Microsoft Office Outlook.
2. Navigate to Inbox, and click the encrypted email.

   Based on your corporate security policies and the access mode that has been configured for you, the *ArcotID PKI Client 6.2 Password* dialog appears and indicates one of the following:

   - You have been granted online access to your credentials, and have to provide your online password.
   - You have been granted offline access to your credentials, and have to provide your offline password.
3. In the Password field, enter the corresponding ArcotID password.
4. Click OK to submit the password.

   The ArcotID PKI Client verifies user's password. If the authentication was successful, then the Microsoft Outlook unencrypts the email.

Working With Adobe PDF Documents

This section describes how to use certificates stored in ArcotID PKI to sign Adobe PDF documents. It covers the following topics:

- Performing Initial Configurations (see page 113)
- Signing Using Private Keys Stored in ArcotID PKI (see page 115)

Performing Initial Configurations

Before using ArcotID PKI to digitally sign PDF documents, users need to configure the Arcot PKCS#11 module with Adobe Reader or Adobe Acrobat, as discussed in this section.

- Integrating with Adobe Reader 8.0 (see page 114)
- Integrating with Adobe Acrobat 8.0 (see page 114)

Integrating with Adobe Reader 8.0

Users must perform the following steps to integrate Arcot PKCS#11 with Adobe Reader 8.0:

Note: The following steps are applicable to Adobe Reader 8.0 *only.* Refer to vendor documentation for information on how to perform these tasks using other versions.

1. Open Adobe Reader.
2. Click the **Document** menu and choose **Security Settings**.

   The *Security Settings* page appears.
3. Expand the **Digital IDs** menu and select **PKCS#11 Modules and Tokens**.
4. In the upper-right pane, click **Attach Module**.

   The *Locate a PKCS#11 Module* dialog appears.
5. Browse to the \Windows\System32 directory. Select the ArcotPK11.dll file and click the **Open** button.

   The Arcot PKCS#11 module appears in the list of modules in the upper-right pane. The **Module Manufacturer ID** must have an entry **Arcot Systems, Inc.**, which ensures that the module has been successfully installed.

Integrating with Adobe Acrobat 8.0

Users must perform the following steps to integrate Arcot PKCS#11 with Adobe Acrobat 8.0:

Note: The following steps are applicable to Adobe Acrobat 8.0 *only.* Refer to vendor documentation for information on how to perform these steps using other versions.

1. Open Adobe Acrobat.
2. Click the **Advanced** menu and choose **Security Settings**.

   The *Security Settings* page is displayed.
3. Expand the **Digital IDs** menu and select **PKCS#11 Modules and Tokens**.
4. In the upper-right pane, click **Attach Module**.

   The *Locate a PKCS#11 Module* dialog appears.
5. Browse to the \Windows**System32** directory. Select the file ArcotPK11.dll and click the **Open** button.
6. The Arcot PKCS#11 module appears in the list of modules in the upper-right pane. The **Module Manufacturer ID** must have an entry **Arcot Systems, Inc.**, which ensures that the module has been successfully installed.

Signing Using Private Keys Stored in ArcotID PKI

The following workflow outlines the steps that users must perform to sign PDF documents by using their certificates stored in the ArcotID PKI:

1. Open the PDF document that has to be signed.
2. Click the Sign menu and select the Place Signature option.
3. Identify the location where the signature has to be placed. Click the left button of the mouse at the identified location and drag the mouse to create a placeholder for the signature.

   The Sign Document dialog appears.
4. In the Digital ID drop-down list, select the digital ID to be used for signing.
5. Click the Sign button.

   The Save As dialog appears.
6. Browse to the location where the PDF has to be saved and click Save.

   Based on your corporate security policies and the access mode that has been configured for users, the ArcotID PKI Client 6.2 Password dialog indicates one of the following:

   - You have been granted online access to your credentials, and have to provide your online password.
   - You have been granted offline access to your credentials, and have to provide your offline password.
7. In the Password field, enter the corresponding ArcotID PKI password.
8. Click OK to submit the password.

   The ArcotID PKI Client verifies user’s password. If the authentication was successful, then the ArcotID PKI Client signs the PDF with the certificate and embeds the user’s digital signature in the PDF.

Chapter 10: ArcotID PKI Client Tools

This chapter describes the following tools used with ArcotID PKI Client:

- ArcotID CSP Registration Tool (see page 117)
- Arcot Offline Tool (see page 118)
- ArcotID Tool (see page 119)

ArcotID CSP Registration Tool

The ArcotCSPInstall.exe file provides the Arcot Cryptographic Service Provider (CSP) tool. It is stored on the user's machine when the Native client is installed. This is a command-line tool, when invoked, it registers the ArcotClient.dll as a CSP within Microsoft Windows CAPI (Cryptographic API). It includes the ArcotClient.dll in its directory and registers it as a CSP using the signature that is stored in the ArcotClient.sig file.

Usage

Registering CSP

Perform the following steps to register Arcot client as a CSP:

1. Open a command prompt window.
2. Navigate to the following directory:
   `<%SYSTEMDRIVE%>:\Program Files\Arcot Systems\ArcotID Client`
3. Enter the following command:
   `ArcotCSPInstall.exe install`

   The ArcotCSPIntall dialog appears.
4. Click **Yes** to register Arcot client as a CSP.

   After the successful registration, the "Arcot Cryptographic Service Provider setup successful" message appears.
5. Click **OK** to continue.

Unregistering CSP

Perform the following steps to unregister Arcot CSP:

1. Open a command prompt window.
2. Navigate to the following directory:
   `<%SYSTEMDRIVE%>:\Program Files\Arcot Systems\ArcotID Client`
3. Enter the following command:
   `ArcotCSPInstall.exe remove`

   The ArcotCSPIntall dialog appears.
4. Click **Yes** to unregister Arcot CSP.

   After the successful unregistration, the "Uninstall succeeded" message appears.
5. Click **OK** to continue.

Arcot Offline Tool

The Arcot Offline Tool (ArcotOfflineTool.exe) is an utility that is installed during the ArcotID PKI Native Client installation. This tool helps users to set offline password that is used for offline access, synchronize ArcotID PKI access, and reset or change offline password. Refer to "Working With Arcot Offline Tool" (see page 104) section for more information on the tool usage.

ArcotID Tool

ArcotIDTool.exe is a command line tool that is not installed as part of the ArcotID PKI Client, but available separately. This tool is used for testing and troubleshooting. It tests the functionality of importing keys and certificates into the *key vault* of an ArcotID PKI. The keys and certificates to be imported must be stored in PKCS#12 or PFX format. The tool is invoked as follows:

Usage

```
arcotidtool ImportP12sWallet arcotid_filename arcotid_PIN
number_of_P12files_toimport P12filename_1 P12password1,
P12filename_2 P12password2,…
```

Where,

- arcotid_filename is the file name of the ArcotID.
- arcotid_PIN is the password of the ArcotID.
- number_of_P12files_toimport is the number of PKCS#12 or PFX files that will be imported into the ArcotID.
- P12filename_1 is the file name of the first PKCS#12 or PFX file.
- P12password1 is the password of the first PKCS#12 file or PFX file to be imported.
- P12filename_2 and P12password2 are the filename and password of the PKCS#12 or PFX file.

Example

```
arcotidtool ImportP12sWallet tooltest1_10_0_21_125.aid 123456 1
MyCerts.p12 123456
```

This command imports the P12 file MyCerts.p12 into the ArcotID tooltest1_10_0_21_125.aid.

Chapter 11: ArcotID PKI Client Compatibility

The ArcotID PKI Client is compatible with previous versions of clients.

Server Support

The ArcotID PKI client will continue to support the necessary set of features to be compatible with any standard AuthMinder Server 5.x, 6.x, and 7.x installations.

ArcotID PKI Support

The client will work with ArcotID PKIs created with previous releases of AuthMinder Server.

New ArcotID PKI format attributes are designed so that an older client can still use the ArcotID PKI for basic purposes such as authentication. The existing 4.x client will not support the new 5.0 card attributes. If new cards are presented, then the client will ignore them without crashing or producing error messages.

New ArcotID PKIs that are protected with device locking (double camouflage; user password and machine PIN) will be ignored by previous clients. This will be done by changing wallet version field in the ASN.1 of the device-locked ArcotID PKI.

New ArcotID PKI files downloaded by the ArcotID PKI Client 5.0 client will not be recognized by any old version of client installed on the same machine, because the ArcotID PKI file extension is changed to .aid.

Appendix A: Source Code Samples

This appendix describes the deployment of ArcotID PKI sample applications and the directory location to examine the files.

Deploying Sample Applications

Perform the following steps to deploy the ArcotID PKI Client:

1. Install WebFort6ClientSample.war on the application server (for example Apache Tomcat), in the following location:

   <*APP_SERVER_HOME*>\webapps

   Note: The deployment procedure depends on the application server that you are using. Refer to your application server vendor documentation for detailed instructions.

2. Restart the application server.

After the WAR file is deployed, the following sample applications are available:

- Login sample with applet:

  http://<Host_name>:<Port_name>/WebFort5ClientSample/loginApplet.html

- Login sample with native client:

  http://<Host_name>:<Port_name>/WebFort5ClientSample/loginActiveX.html

- Login sample with flash client:

  http://<Host_name>:<Port_name>/WebFort5ClientSample/loginFlash.html

- Login sample that detects multiple clients and attempts to use the best available.

  http://<Host_name>:<Port_name>/WebFort5ClientSample/login.html

The following URL is used to configure the URL of the AuthMinder Server:

http://<Host_name>:<Port_name>/WebFort5ClientSample/setup.html

The following URL is used to *upload* a test ArcotID PKI for use with the samples. The ArcotID PKI must be issued by the same AuthMinder installation that is used for the authentication.

http://<Host_name>:<Port_name>/WebFort5ClientSample/uploadArcotID.html

Code Samples

The following directories containing the sample source code.

- /WebFort5ClientSample

  Contains the home HTML file for each sample application.

- /WebFort5ClientSample/client

  Contains the Arcot applets and flash client used by the sample applications.

- /WebFort5ClientSample/images

  Contains image files used by the sample applications.

- /WebFort5ClientSample/js

  Contains JavaScript files used by the sample applications.

- /WebFort5ClientSample/jsp

  Contains JSP's used by the sample applications.

- /WebFort5ClientSample/WEB-INF

  Contains classes used for the servlets that enable the sample applications to communicate with the AuthMinder Server (get challenge, verify signed challenge.)